Applied Intelligence
https://doi.org/10.1007/s10489-022-03235-7

Hashing-based semantic relevance attributed knowledge graph embedding enhancement for deep probabilistic recommendation

Nasrullah Khan[1,2] · **Zongmin Ma**[1,3] · **Li Yan**[1] · **Aman Ullah**[4]

Accepted: 11 January 2022

Abstract
Knowledge graph embedding (KGE) is effectively exploited in providing precise and accurate recommendations from many perspectives in different application scenarios. However, such methods that utilize entire embedded Knowledge Graph (KG) without applying *information-relevance regulatory* constraints fail to stop the noise penetration into the underlying information. Moreover, higher computational time complexity is a CPU overhead in KG-enhanced systems and applications. The occurrence of these limitations significantly degrade the recommendation performance. Therefore, to cope with these challenges we proposed novel KGEE (Knowledge Graph Embedding Enhancement) approach of *Hashing-based Semantic-relevance Attributed Graph-embedding Enhancement* (H-SAGE) to model semantically-relevant higher-order entities and relations into the unique Meta-paths. For this purpose, we introduced *Node Relevance-based Guided-walk* (NRG) modeling technique. Further, to deal with the computational time-complexity, we converted the relevant information to the Hash-codes and proposed *Deep-Probabilistic* (dProb) technique to place hash-codes in the relevant hash-buckets. Again, we used *dProb* to generate guided function-calls to maximize the possibility of Hash-Hits in the hash-buckets. In case of Hash-Miss, we applied *Locality Sensitive* (LS) hashing to retrieve the required information. We performed experiments on three benchmark datasets and compared the empirical as well as the computational performance of H-SAGE with the baseline approaches. The achieved results and comparisons demonstrate that the proposed approach has outperformed the-state-of-the-art methods in the mentioned facets of evaluation.

Keywords Knowledge graph · KGEE · Recommendation · Information relevance · Hashing · DNN

1 Introduction

With the current burst of data-overload on the web, choosing feasible options in the bulks of analogous choices is challenging. The process of streamlining the online data-overload and suggesting relevant options to the end users to satisfy their needs and interests is personalized recommendation. Recommender Systems (RS) exploit the information of individuals and their concerned people to provide suggestions about their potential interests [1, 2]. Previously, the interaction information was acquired from the user's previous interaction-logs on the web and preferences about their hidden interests were generated via CF[1]-based techniques [3] – the most common and widely utilized recommendation algorithms [4]. Although these methods achieved great importance and consideration, their performance was significantly affected by the issues of data-sparsity, gray-sheep and cold-start [5, 6]. To cope with these challenges, researchers

[1] Collaborative Filtering

✉ Zongmin Ma
zongminma@nuaa.edu.cn

Nasrullah Khan
nasrullah@nuaa.edu.cn

Li Yan
yanli@nuaa.edu.cn

Aman Ullah
dr.aman@csu.edu.cn

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

[2] Department of Computer Science, COMSATS University Islamabad, Vehari 61100, Pakistan

[3] Collaborative Innovation Centre of Novel Software Technology and Industrialization, Nanjing 210000, China

[4] School of Computer Science and Engineering, Central South University, Changsha 410083, China

Published online: 06 May 2022

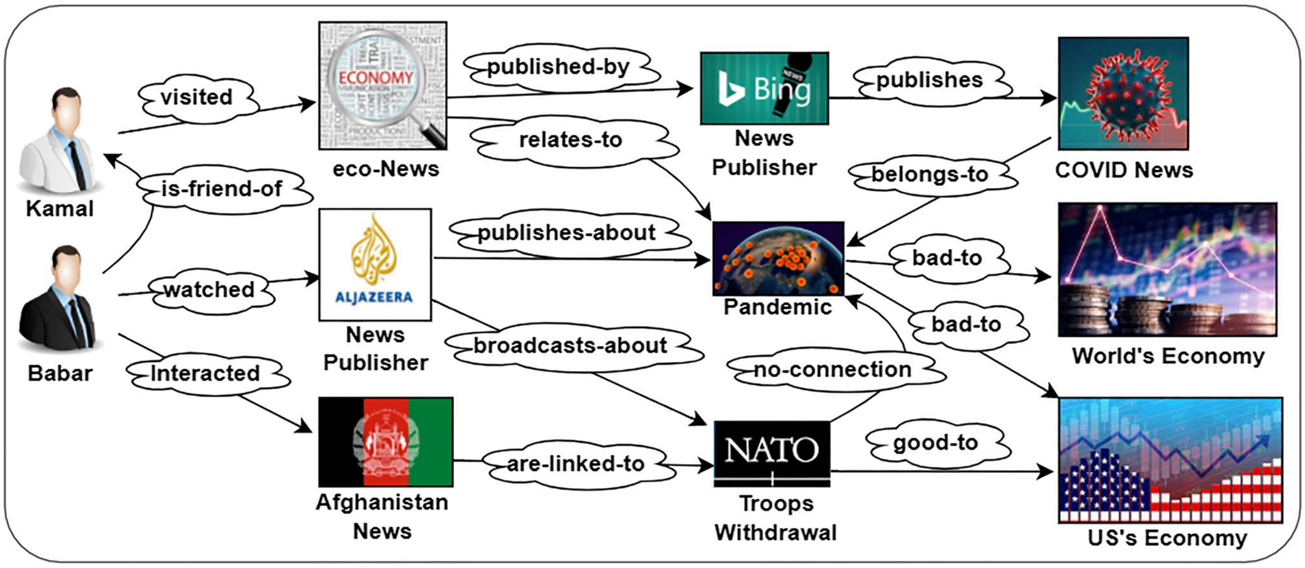


Fig. 1 News Recommendation's – an exemplary scenario; where Babar interacts with *"Aljazeera News"* and *"Afghanistan News"* whereas *"Kamal"* visited *"eco-news"* published by *"Bing-News"*. In response, *"Covid-News"* and *"Economy News"* are recommended to both of them

adapted to utilize side information acquired from different knowledge resources as input to the recommender systems. Although the side information exploited by the recommendation methods has many types, KG-based side information gained valuable attention and popularity. It describes information via entities and relations among them and exhibits easy theme of manipulation, processing and understanding.

The information structure of Heterogeneous Information Graph (HIG) can easily be modelled into feature vectors [7, 8]; and this ideology is highly empowered by the exploitation of language-modeling techniques with graph theory. Previously, inter-connections among words in the documents (e.g., like relations among nodes in KG) was introduced via pairwise-interaction in the vector space. Similarly, based on the distance among vectors in the Euclidian space, NLP[2]-research defined distance-based language modelling to represent the structured-embedding of triplets. Moreover, translation-based models represent relations among entities as translation in vector space and assign preferential values to the entities based on the distance among them via distance function. NLP-based language-models that support random-walk strategies over the highlighted *"words in the documents"* up to 1st-order relation, proposed idea about the possibility of sequential flow among nodes via relations. Later, random-walk *"in the documents"* up to the 2nd-order relation was developed which introduced the concept of walk over nodes in KG.

[2] Natural Language Processing

To efficiently exploit KG-based side information via systems and application domains, entire KGs are embedded to the low dimensional Euclidian space – the process is termed as KGE. It is basically vector-notion of nodes and paths that preserves the semantics and topologies of KG data in the embedded state. KGE is significantly contributing to RS by providing the data in feasibly-interlinked-mode of presentations, i.e., KG-based side information [9, 10]. RS utilize KG-based side information to provide personalized recommendations based on item-relations, user-item-interactions and user-feedback.

Although *"paths"* are information channels among nodes in the KG to infer reasoning-facts for recommendation, suffer from constraints and limitations wrt the implementation concerns. To well demonstrate the scenario, we picturize a tiny exemplary situation from KG-based news recommendation, as shown in Fig. 1. Let consider path *"P1: Babar → Kamal → eco-News → News Publisher → Covid News → Pandemic → World's Economy"*. A length constraint *LC* has to be imposed on *"P1"* that results in truncation of nodes exceeding the length limit of *LC*. For instance, *"→ Covid News → Pandemic → World's Economy"* will be discarded from *"P1"* if $LC = 3$. Moreover, linear paths fail to reflect the graph structure in information aggregation, and a CPU overhead in processing all of the outgoing paths on *"Babar"* without having a concern of information relevance. Similarly, the current methods that aggregate information from neighbors of the entities regardless of applying information relevance constraints aggregate noise with information. Let see how noise penetrates into the information being aggregated; e.g., if everything in under-consideration part of KG is irrelevant except this meta-path, *"Kamal → eco-News → Pandemic → US's Economy"*, then all of the aggregated information coming-in and going-out via the node *"Pandemic"* is noise.

Subsequently, to effectively deal with the above discussed research challenges, we proposed a novel hashing-based semantic relevance attributed Knowledge Graph Embedding Enhancement (KGEE) approach for effective recommendations. The framework contains Relevance-based Influential-graph Construction (RIG) and Hashing-based Recommendation (HR) modules. The data (i.e., KG-in-raw and user interaction-logs) is given as input to the RIG module. RIG captures higher-order semantically relevant entity-information via the connecting relations, and HR transforms the corresponding information to the hamming space to generate recommendations. To the best of our knowledge, KGEE is a first ever contribution to the hashing-based recommendation in the scope of heterogeneous data. The experimental-work and theoretical-comparison confirm that H-SAGE has outperformed the state-of-the-art methods with significant improvements.

The key contributions of this study are given below;

- We proposed *Node Relevance-based Guided-walk* (NRG), i.e., path-modeling technique, to highlight Unique Meta Path (UMP) based on semantically relevant entities in KG. NRG works like the spreading effect of an influential-graph.
- We converted UMPs to hash-codes and placed the corresponding hash-codes in semantically relevant hash-buckets-based on their mutual likelihood to maximize the possibility of Hash-Hits via function calls.
- In case of Hash-Miss, we used LS-hashing to acquire the required hash-codes around the location of Miss up to a maximum length of 3×3 hash-indices. Upon success, we return the required hash-codes to the function, otherwise we return 0.
- We exploited a predictive presentation interface that collects the retrieved hash-codes, processes them, calculates the potential preferences and generates the formal recommendation responses.
- We performed extensive experiments on three real world benchmark datasets to assess and analyze the performance of H-SAGE in comparison with the baseline methods. The experimental results and theoretical comparison demonstrate that the proposed approach has outperformed the baseline methodologies.

Further organization of this paper is as follows; Section 2 covers the Related Works, Section 3 describes the Preliminaries, Section 4 presents The Proposed Methodology, Section 5 demonstrates the Experimental, Empirical & Theoretical comparative analysis; and finally, Section 6 concludes this paper with Conclusion and Future-work.

2 Related works

Although current KG-based recommendation undergoes various considerable methods in particular, we broadly categorize the relevant work in three classes from the perspectives of implementation techniques, i.e., *Path*, *Embedding and Propagation*, and *Hashing*-based recommendation methods.

2.1 Path-based recommendation

Path-based methods exploit similarity of the interaction-sequence of previous occurrences to provide new recommendations. For example, PER [8] learned user to item and item to item relations through feature extraction with Meta-path-based walk. It is basically HIG-based entity recommendation approach. Typically, it introduced hidden features via meta-paths to demonstrate the relations among entities (i.e., users and items) wrt different connectivity-types in the data. Formally, it modelled HIG-based data differently wrt different users to provide quality recommendations based on user-to-item implicit interactions and defined recommendation models. SemRec [11] introduced the concept of weighted information structure (i.e., HIG and meta-paths) to demonstrate the semantics of relations via differentiating the values of different link-attributes. Further, it predicted user-to-item potential preference (i.e., rating scores) via its proposed semantic meta-path-based recommender system. FMG [12] calculated similarities of path-sequences between users and items in KG wrt the flow of previous interactions and generated new recommendations. Similarly, MCRec [13] utilized meta-path-based random connectivity-processing to obtain the representations of user-to-item context. It combined Deep Neural Network (DNN) with Attention Mechanism (AM) to exploit the rich information context of HIG through Meta-paths for top-N recommendation. It applied priority-guided path sampling technique to choose quality-path samples from the context to construct meta-paths.

KGR [14] analyzed reader's exploration history to model his research interest, and extracted the required material from different research proposals. KGR considered the distance between the reader's research interest and his required information as the knowledge gap or path. Correspondingly, HERec [15] learned the entity embedding representations via the Meta-path-based random-walk sampling technique. Although these methods attained significant performance and popularity in the context of path matching, their major drawback is being fully dependent on manual or random selection of the meta-paths. As a solution to the problem,

[16] proposed RKGE approach to learn semantical representations of users, items and relations among them to forecast the preferences of users towards items of their potential interest. Typically, it applied Recurrent Neural Network (RNN) to sample the semantical context of the paths connecting same or identical entity pairs. Similarly, KPRN [17] exploited neural networks to automatically mine the required meta-paths of the defined length.

However, capturing user to item graphical structure via independent, limitedly-lengthened and linear meta-paths; and computing path-based static similarities to retrieve the potential preferences lead to the wastage of notable extent of information and unwanted CPU overhead respectively.

2.2 Embedding and propagation-based recommendation

KG-based information is embedded to the low dimensional vector space via translation techniques, enriched through mapping to the relevant entities in the external Knowledge Bases (KBs) and tackled through the recommendation techniques for recommendation. For instance, CKE [18] aggregated CF-technique with item-based side information in a Bayesian network to acquire the semantic embedding of items via TransR. KSR [19] accomplished the sequential recommendations via TransE, and DKN [20] acquired the representations of entities-and-relations embedding based on KG-features-learning via TransD. Similarly, RCF [21] used DistMult and attention mechanism to access item-to-item relations for preference computation. Typically, it utilized various types of item-relations for recommendation. It proposed that relation-type (i.e., Allen Turing) and relation-value (i.e., Turing Machine) both are significant for recommendation and greatly impact the performance wrt the preference calculation. KTUP [22] used TransH to jointly learn the recommendation and KG completion modules without preserving the semantic connections among the data instances. RippleNet [23] propagated the historical interactions of users to items over the graph and aggregated the potential preferences of users about unseen-items of their interest based on the propagated and aggregated information. KGCN [24] incorporated the information of neighbors of items into the neural network to learn the embeddings of items with GCN[3] via propagation to calculate the preferences of users.

Correspondingly, KGAT [25] enriched the embeddings of users and items, and recursively performed the information propagation over the graph to enhance the performance; and AKGE [26] applied the Euclidian-distance-based similarity technique to filter out the irrelevant information, constructed local subgraph with relevant entities, and performed relation aware propagation over the network to enhance the performance. Moreover, NACF [27] also identified the potential interests of the users via preference propagation based on the information collected from the neighbors of the entities. DKEN [28] highlighted the impact of information-exchange between the implicit-interactions and explicit-semantics in user-to-item interactions and KG-features respectively to acquire a better grip on semantical and hierarchical structure of information-flow in the graph. Regularization-based approaches utilized the graphical structure of underlying data to acquire the entity representations via accessing the regularization terms; and Unification-based approaches combined the regularization terms with path-based methods to enhance the recommendation performance.

Although the majority of these methods attained great focus and consideration due to their overwhelming performance, noise-free and effective undertaking of the semantical and hierarchical structure of KG-based relevant information via the recommendation algorithms is nevertheless a challenge.

2.3 Hashing-based recommendation

Hashing techniques are exploited to reduce the computational time complexity of algorithmic processes in different application scenarios [29]. Currently, to make data management efficient and smoother, KG-based information processing is being notably incorporated in many domains of computational intelligence. Although KG retains powerful theme of information presentation and management, it undergoes quadratic and higher computational time complexity in normal cases. Unsurprisingly, KG-based side information is utilized for recommendation generation in particular, and hashing-enhanced techniques are used to reduce the computational time complexity of the recommendation as well.

In this section, therefore, we enlist such relevant approaches that exploited hashing techniques in recommendation and information retrieval to lighten the computation. For instance, [30] proposed semantic hashing method to deal with query and document-based textual data to retrieve the required information, and [31] proposed hash-graph-kernel-based approach to extract the interaction of protein-to-protein from the context of node-neighboring in the graph. Similarly, HashNet [32] proposed deep-learning-based approach to use hashing-by-continuation with convergence to learn the exact binary-codes from the imbalanced experimental data. Moreover, [33] introduced KG2Rec to exploit the Locality-sensitive Hashing [34] though CF-algorithms for KG-enhanced recommendation. NeuHash-CF [35] proposed their approach of content-based neural hashing via CF-algorithms to address the limitations of cold-start in recommendation. RSLH [36] tried to reduce the length of hash-codes to the minimal possible extent of applicability via the exploitation of reinforcement

[3] Graph Convolutional Network

learning. Also, HashGNN [37] proposed GNN[4]-based deep-hashing approach to accomplish recommendations with KG. Last but not least, HALF [29] applied search oriented KGE-technique via hash-learning framework to enhance time-complexity-based performance of computation.

The above discussed hashing-based approaches achieved better performance in their applied circumstances to improve the computation against time-complexity, but these methods have overlooked the issues of noise inclusion (emergence, existence) to/in the experimental data. Also, they did not apply any information relevance check or noise filtration constraint to verify (i.e., validate) the applicability-health of the underlying data. On the other hand, our main objectives are to keep the irrelevant data excluded from the underlying information (i.e., experimental datasets), and to perform KG-enhanced recommendation in a lighter-environment of computation. For this purpose, therefore, we propose novel hashing-based semantic relevance attributed KGEE approach of computationally less-heavier recommendation over optimally noise-free KG-data.

3 Preliminaries

In this section, we define primary notations, recommendation task and some of the basic concepts.

3.1 Notations

Set of entities $\mathcal{E}=\{e_1,e_2,\dots,e_k\} \mid k=|\mathcal{E}|$, set of users $\mathcal{U}=\{u_1,u_2,\dots,u_l\} \mid l=|\mathcal{U}|$, set of items $\mathcal{V}=\{v_1,v_2,\dots,v_m\} \mid m=|\mathcal{V}|$ given that $\mathcal{U},\mathcal{V}\in\mathcal{E}$, and set of relations $\mathcal{R}=\{r_1,r_2,\dots,r_\rho\} \mid \rho=|\mathcal{R}|$ belongs to KG. Entities belong to Entity matrix E as $e_j\in E \mid j=1$, 2, …, k, and relations r belongs to Relation matrix R as $r_j\in R \mid j=1$, 2, …, k, given that, $e_j=[n_{ij}]\in\{\pm1\}^\mu$ and $r_j=[p_{ij}]\in\{\pm1\}^\mu$, and correspondence of ith instance with jth entity or jth relation represents the node n_{ij} or path p_{ij}, respectively, belongs to μth dimension of the feature vector. The *interaction* $\mathcal{I}$ among entities is handled through 1 or 0 in *interaction-matrices* $\mathcal{M} \mid \mathcal{M}\in\mathbb{R}^{k\times d}$ as if there exists an interaction or ith instance is similar to jth entity or jth relation, the entry of node $n_{ij}=1$ or path $p_{ij}=1$, respectively, and 0 else to the corresponding matrix. The *neighbors* of node n are shown as $\mathcal{N}(n)$. Moreover, anchor-points γ belongs to the training set X and placed in Anchor matrix A as $\gamma_i\in A \mid i=1$, 2, …, υ, where the items of A are randomly selected from X. Similarly, h belongs to Hash matrix $H \mid H\in\{\pm1\}^{l\times k}$ as $h_i\in H \mid i=1$, 2, …, l, $\|H\|_2$ is normalization of H, and $H^\top$ is transpose of H. Thus, H_e and H_r are the hash matrices of the corresponding E and R matrices respectively.

[4] Graph Neural Network

3.2 Problem description

This approach is supposed to produce a list of top-K recommendations out of a bulk of candidate items. Therefore, it scans raw-data and interaction-log $\mathcal{I}_\ell$ as inputs, exploits the relevance factor and $\mathcal{I}_\ell$ to filter-out the irrelevant data, and utilizes relevant information to construct the influential sub-graph $\mathcal{G}_s$. The $\mathcal{G}_s$ is transformed to hash-codes to forecast the probability of u_i selects v_j via $\hat{\delta}_{(u_i,v_j)}=\varphi\left[e_{u_i,v_j}\otimes C_\mathrm{H}\right]$; where $\varphi \mid \varphi(e_h,r,e_t)\in\{\mathcal{E}\times\mathcal{R}\times\mathcal{E}\}$ is the prediction function that predicts true likelihood among the entities of the unseen triplets (e_h,r,e_t), $\bigotimes$ is the set of environment parameters, e is entity and C_H represents the hash-codes. The descending-order sorted outcome of $\hat{\delta}_{(u_i,v_j)}$ illustrates the list of top-K recommendations.

3.3 Definitions

In this section, we define some technical terminologies that are to be utilized in implementation or comparison.

3.3.1 Hash function

A function f is hash function f_h if $f_h:\Upsilon^\mu\to\Upsilon^{\alpha(\mu)} \mid f_h=|\mu| \wedge f_h\in\mathcal{F}_\alpha$ where $\Upsilon=\{-1,1\}$ satisfies the polynomial bounded predetermined constraint $\alpha:\Gamma\to\Gamma \mid \alpha(\mu)<\mu,\forall\mu\in\Gamma$, and belongs to the infinite family $\left[\mathcal{F}_\alpha\right]_{\alpha=1}^\infty$ of functions. A f_h is collision-proof if it map information instances from a non-fixed bulk of triplets into a fixed set of independent data entities with an expected ideal hit-ratio of $A\leftarrow f_h(B) \mid f_h(A)=A$ optimally for all cases, but it is NP-hard task.

3.3.2 Information hashing

It is the process of transforming data-instances to the hamming space, i.e., $\{-1,\ 1\}^\mu$ where $\mu<|\mathcal{E}|$, to learn non-linear mapping $f_h:y\longrightarrow h\in\{-1,\ 1\}^\mu$ to transform each entity $y\in\mathcal{E}$ into a μ-bit binary hash-codes $h=f_h(y)$, such that; the relevance among the transformed entities can be preserved in hash-codes.

3.3.3 Hamming-space constraints

Hamming space imposes *un-correlation* and *balance* as validation constraints on the bits of transformed information. Un-correlation and bit-balance refer to *the sparseness of vector's dimensions* and *equal probability of hit or miss* respectively. Specifically, un-correlation is *the rows must not have any correlation with other rows in the corresponding matrix*, and bit-balance *is out of any two bits of 0 and 1, we must have a 1.*

3.3.4 Locality sensitive (LS) hashing

The LS function depends on *the distance between two consecutive data-points* and *the probability of collision of the data-points is inversely proportional to their mutual distance*. LH belongs to the family of hash functions f_h that maps a location α from testing information instances to retrieve a location β from z hash-bucket (HB) as HB_z in the bucket-space Ω as $LS = \{z : \alpha \rightarrow \beta\}$. Formally, to retrieve β from HB_z via a function call f_c, the following condition defines LS as:

$$f_c = \begin{cases} \mathrm{P}\left[\mu_h(\alpha) = \mu_h(\beta)\right] > \alpha_{h+1}, \text{if } \beta \in \Phi(\alpha, y) \\ \mathrm{P}\left[\mu_h(\alpha) = \mu_h(\beta)\right] < \alpha_{h+2}, \text{if } \beta \notin \Phi(\alpha, y) \end{cases} \tag{1}$$

Where $\mathrm{P}[\cdot]$ is probabilistic function and $\Phi(\alpha, y) = \{\beta | \Omega(\alpha, \beta) \le y\}$ is a rectangular-shaped lengthy box having β as a target location of the projected location α, and † shows the number of possible attempts (i.e., $y = 3$) to get the required location in the case of miss. Moreover, $\alpha \simeq \beta$ is assumption of hit, α_{h+1} and α_{h+2} are constants; $h = 0$, $y > 0$ and $0 < \alpha_{h+1} < \alpha_{h+2} < 1$. The probability of collision of α and β decreases with each increase in the distance between them in Ω.

3.3.5 Knowledge graph and Meta-path

A graph $G = (E, P, \Phi, \Psi)$, where E represents the set of entities, P shows the set of paths-between-entities, Φ denotes the entity-type mapping function f_e as $\Phi_{f_e} : E \longrightarrow \mathcal{E}$, and Ψ describes the path-type mapping function f_p as $\Psi_{f_p} : P \longrightarrow \mathcal{R}$. Therefore, each entity $e \subseteq E$ is mapped to the specific entity-type in E, i.e., $\Phi_{f_e}(e) \in \mathcal{E}$, and each path $p \subseteq \mathcal{R}$ is mapped to the concerned path-type in $\mathcal{R}$, i.e., $\Psi_{f_p}(p) \in \mathcal{R}$. If $|\mathcal{E}| > 1$ or $|\mathcal{R}| > 1$ then G is heterogeneous. Meta path $\wp$ is a sequence of finite paths p between the consecutive nodes in G. For instance, a Meta path from n_a to n_d is written as $\wp(n_a, n_d) = p_1 \circ p_2 \circ p_3$ implies that $\wp \Rightarrow n_a \xrightarrow{p_1} n_b \xrightarrow{p_2} n_c \xrightarrow{p_3} n_d$, where $\circ$ is the composition operator.

3.3.6 Interaction log and session

Interaction log $\mathcal{I}_\ell$ is a timespan-based series of user-interactions preserved through user profiling [38] and stored in $\mathcal{I}_\ell$ tracks. The $\mathcal{I}_\ell$ contains set of items $\mathcal{V} = \{v_1, v_2, \dots, v_m\}$, set of interactions $\mathcal{I} = \{i_1, i_2, \dots, i_\psi\}$ like viewed, rated, purchased, etc., set of sessions $\mathcal{S} = \{s_1, s_2, \dots, s_\psi\}$, and set of timespans $T = \{t_1, t_2, \dots, t_\psi\}$. For instance, an arbitrary session $s_j = (v_j, i_j, t_j) \mid s_j \in \mathcal{V} \times \mathcal{I} \times T$ or $s_j = (v_j, i_j) \mid s_j \in \mathcal{V} \times \mathcal{I}$ where T is implicitly retrieved from sessions $\mathcal{S}$ or explicitly collected from interactions $\mathcal{I}$ respectively.

4 The proposed methodology

This section contains Pre-Processing, Hashing and Presentation modules. In this section, we purify the datasets, construct the influential (local subgraph) graph, perform Hashing, and compile-&-display the recommendation outcomes.

4.1 Pre-processing module

Data Embedding, *Dimension Rationalization*, *Data Purification* and *subgraph Construction* are described in this module.

4.1.1 Data embedding

We used TransD [39] to embed entities and relations to the independent vector spaces and preserved their isolation through the following precise matrices

$$\mathcal{M}_{p,h} = \overline{p} \cdot \overline{n}_h^{\top} + \mathcal{I}^{k \times d} \tag{2}$$

$$\mathcal{M}_{p,t} = \overline{p} \cdot \overline{n}_t^{\top} + \mathcal{I}^{k \times d} \tag{3}$$

Where h, p, t denote a triplet, and $\overline{p}$ and $\overline{n}$ show the embedded information; entities belong to $\mathbb{R}^d$ and relations to $\mathbb{R}^k$, $\mathcal{I}^{k \times d}$ represents the identity matrix having 1 s on diagonal and 0 s elsewhere, and $\mathcal{M}_{p,h}$ and $\mathcal{M}_{p,t}$ are used to project the vectors of head and tail entities to concerned relation-spaces respectively. Subsequently, the plausibility factor f_p of a given triplet (n_h, p, n_t) is defined as

$$f_p(n_h, p, n_t) = -\left\| \mathcal{M}_{p,h} \cdot n_h + p - \mathcal{M}_{p,t} \cdot n_t \right\|_2^2 \tag{4}$$

4.1.2 Dimension rationalization

Although placing identical information instances in identical vector spaces is important for efficient computation, the initial embeddings span over vast vector dimensions, and filtering out the less relevant instances is a possible solution [40]. Therefore, to filter out the less relevant data-instances we introduced a Relevance-Factor *Rel* constrained by the threshold ϑ. *Rel* evaluates the relevance-extent $Rel_{c,j}$ between current node n_c and the candidate node n_j to validate whether to discard n_j or not. If the candidate of $Rel_{c,j}$ satisfies ϑ, that is assigned with 1 and 0 otherwise. All entries with 1 are orthogonally stored via Eq. (2) or (3) and rest of the entries are discarded and updated with 0 according to Lemma 1, based on their relative relevance-extents with n_c. Therefore, the transition matrix is linearly computed because further it doesn't depend upon the previous entries.

Lemma 1 *Mutually relevant data instances are orthogonally co-related in nature and present a unit matrix $k \times l$ in the Euclidian embedding space.*

Proof: Eq. (2) and (3) represent ordinary matrices that can be operated wrt different matrices for storage or transformation. Therefore, we transform the information instances n^k to the orthogonal basis $\{n^k\} = \left(n^k_l, n^k_l, \dots, n^k_L\right)$ of orthogonal basis matrix $\mathcal{M}_{\mathcal{O}} = \left(n^k_i\right)$ to maintain a sequential flow among them. Let's recall from the basic matrix theory, for each i of $\mathcal{M}_{\mathcal{O}}$, the first i instances n^i form orthogonal base from $[k,l] \mid k=l=0$ to $[K,L]$ wrt the winding instance of $\{n^k\}$, is expressed as $n^k = n^k_l$ by increasing $[k,l]$ by 1 with each iteration; where k and l increase with an increase in row and column respectively. The highest index value of k and l in the matrix depends upon the row-head index of k.

Henceforth, $i+1$ instances (i.e., n^{i+1}) creates cosine angles with i instances n^i from $[k+2, l+2]$ to $[K,L]$ wrt $\{n^{k+1}\}$ as $sim_{i,\,i+1} = \cos(n^{k+1}, n^{k+2})$. Since, the underlying instances have unit length; thus, their cosine is equal to their dot product as $\cos(n^{k+1}, n^{k+2}) = n^{k+1} \cdot n^{k+2}$, by putting $n^k_l = 0$, $\forall l > i$, we formulate the expression of similarity between two instances as $n^{k+1} \cdot n^{k+2} = \sum_{i=0}^{k} n^{k+1}_{l+1} n^{k+2}_{l+2}$, where n^{k+1}_{l+1} is the first instance of i, and thus $\omega_l = n^{k+1}_{l+1}$. As, we already mentioned that the instances have unit lengths, thus, the length of n^{k+1} is equal to 1. Thus, $n^{k+1} \cdot n^{k+2} = \sum_{i=0}^{k} n^{k+1}_{l+1} n^{k+2}_{l+2} = \sum_{i=0}^{k} \left(n^{k+i}_{l+i}\right)^2 = 1$.

$$\begin{bmatrix} n^{k+1} \\ n^{k+2} \\ n^{k+3} \\ n^{k+4} \\ n^{k+5} \\ \vdots \\ n^K \end{bmatrix} = \left[\begin{array}{ccccccc} & & & & & \cdots & \\ \boldsymbol{n^{k+1}_{l+1}} & 0 & 0 & 0 & 0 & \cdots & 0 \\ n^{k+2}_{l+1} & \boldsymbol{n^{k+2}_{l+2}} & 0 & 0 & 0 & \cdots & 0 \\ n^{k+3}_{l+1} & n^{k+3}_{l+2} & \boldsymbol{n^{k+3}_{l+3}} & 0 & 0 & & 0 \\ n^{k+4}_{l+1} & n^{k+4}_{l+2} & n^{k+4}_{l+3} & \boldsymbol{n^{k+4}_{l+4}} & 0 & \cdots & 0 \\ n^{k+5}_{l+1} & n^{k+5}_{l+2} & n^{k+5}_{l+3} & n^{k+5}_{l+4} & \boldsymbol{n^{k+5}_{l+5}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ n^K_{l+1} & n^K_{l+2} & n^K_{l+3} & n^K_{l+4} & n^K_{l+5} & \vdots & \boldsymbol{n^K_L} \\ & & & & & \cdots & \end{array}\right] \tag{5}$$

Since, for index $[1,1]$ in Eq. (5), n^{k+1}_{l+1} is the first instance of n^{k+1} wrt i, and hence $\omega_l = n^{k+1}_{l+1}$. Let us consider $n^{k+1}_{l+1} = 0$ for all $l > k+1$ and $\alpha = k = l$, then from $\omega_\alpha = n^{\alpha+1}_{\alpha+1}$; for the previous instance of k, we have $\omega_{\alpha+1} = 1$ as

$$\omega_{\alpha+1} = \sqrt{1 - \sum_{i=0}^{k} n^k_l n^{k+1}_{l+1}} = \sqrt{1 - \sum_{i=0}^{k} \left(n^{k+1}_{l+1}\right)^2} \tag{6}$$

Hence it is proved that for all instances, $\omega_{\alpha+1}$ effectively describes the coordinates of i against l wrt $\{n^k\}$. On substitution of 1 against $k=l$ in Eq. (5), we have $\left[n^1\right] = \left[\,1\;0\;\;0\;0\;\dots\,\right]$, $\left[n^2\right] = \left[\,0\;1\;\;0\;0\;\dots\,\right]$, $\left[n^3\right] = \left[\,0\;0\;\;1\;0\;\dots\,\right]$ and so on. Further, we also described the raw iteration set wrt all of the related entries in corresponding matrix as shown in Eq. (5).

4.1.3 Data purification and influential-graph construction

We introduce *Node Relevance-based Guided-walk* (NRG) modeling technique to refine data by determining, highlighting and maintaining the prominent paths in the embedded space. Although a large quantity of works exist in the literature [14, 15] on meta-path-based random crawl sampling techniques, it face diverse types of limitations that significantly degrade the performance of recommendation. For instance, path length constraints, inability to effectively capture the graph structure due to linear paths, difficulties in optimized path selection, etc. are a few notable examples of their drawbacks. But the major drawback is applying no-constraint on information relevance concern; all cadres of the data is considered as the part of the underlying information that result in performance degradation and unwanted computational overhead. Therefore, we apply NRG modeling technique to determine such unique meta-paths that portray semantically relevant information to construct the influential-graph $\mathcal{G}_s$. The modeling mechanism of NRG is similar to the spreading effect of a pandemic. The main objectives of NRG modeling technique are; (*i*) Determining semantically relevant nodes, (*ii*) Maintaining single hop prominent paths, (*iii*) Unifying single hop prominent paths together to create meta-paths, (*iv*) Identifying the meta-paths via unique identifiers (IDs), (*v*) Interlinking the meta-paths wrt their IDs to create $\mathcal{G}_s$, (*vi*) Discarding the irrelevant data.

Using Eq. (7) and (8), we randomly consider a node as a central node $n_{\mathcal{C}}$, termed it as the current node $n_c \mid n_c = n_{\mathcal{C}}$, calculate the relevance between n_c and the candidate nodes $n_j \mid j = 1, 2, 3\dots$, and selected the node having the highest similarity with n_c as the target node n_t for the next step of the walker. Then for the second hop, we again consider n_t as $n_c \mid n_c = n_t$ and repeated the process to get the next relevant node, and so on. After completing a 5-hops long meta-path and assigning it with the identifying ID, e.g., $\wp_1$, we come back to the initial point, i.e., 1st-hop, and consider the node having the second highest relevance with n_c, and repeated the process for the next meta-path. After evaluating all of the neighbors of n_C, we considered the highest-relevant neighbor of the n_C as the next n_C and repeated the process. Iteratively, the model performed the same technique for all of the nodes, and subsequently, the meta-paths were stored in an indexed track wrt their assigned IDs, as described in Algorithm 1. Finally, the meta-paths are combined based on their IDs to construct the local subgraph.

We calculated the relevance between the corresponding nodes in two steps of comparison and sum-up the outcomes; (1) tendency between n_c and n_j, (2) Local similarity between n_c and n_j, and the similarity between n_c and the information on the path going from n_c to n_j.

For (1), we calculated the factor of similarity-based node-to-node tendency to predict the next node, as:

$$Ten_{Sim}\left(n_c, n_j\right) = \frac{\left[\left(\mathcal{N}\left(n_c\right) \cap \mathcal{N}\left(n_j\right)\right) + 1\right] + f(ten)_{n_c,n_j}}{\left[\left(\mathcal{N}\left(n_c\right) \cup \mathcal{N}\left(n_j\right)\right) + 1\right] * \sum_{\forall\Gamma|\Gamma\in\mathcal{N}\left(n_c\right)} f(ten)_{n_c\Gamma}} * 100 \tag{7}$$

Where $f(\cdot)$ shows the function that calculates the tendency between the concerned nodes, defined as:

$$f(ten)_{n_c,n_j} = x \cdot \mathcal{I}_r(ten) + (1-x).\mathcal{I}_p(ten) \tag{8}$$

Where $x \mid 0 < x < 1$ and *ten* represent the co-efficient and parameter of tendency; $\mathcal{I}_r$ and $\mathcal{I}_p$ show the recent and previous user-interactions, respectively. In the case of tendency, the *ten* factor greatly depends upon the input of $\mathcal{I}_\ell$.

In case of (2), for (n_c, n_j) and (n_c, n_j, p), we apply Adamic and Jaccard (AJ) similarities, respectively:

$$Sim_{AJ}\left(\left(n_c, n_j\right) + \left(n_c, n_j, p\right)\right) = 0.25 * \left(\sum_{y\in\left(\mathcal{N}\left(n_c\right)\cap\mathcal{N}\left(n_j\right)\right)} \frac{2}{\log|\mathcal{N}(y)|} + 2 * \mathrm{X}\right) \tag{9}$$

Where $\mathrm{X} = \frac{\left[\left(\mathcal{N}\left(n_c\right)\cap\mathcal{N}\left(n_j\right)\right)+1\right]*|p|}{\left[\left(\mathcal{N}\left(n_c\right)\cup\mathcal{N}\left(n_j\right)\right)+1\right]*|p|}$ represents the ratio of common to all neighbors of the candidate node.

To get the required relevance between n_c and n_j, we summed up Eq. (7) and (8) into (10) as,

$$Rel\left(n_c, n_j\right) = Ten_{Sim}\left(n_c, n_j\right) + Sim_{AJ}\left(\left(n_c, n_j\right) + \left(n_c, n_j, p\right)\right) \tag{10}$$

Where *Rel* describes the extent of the relevance between the current and the candidate nodes. The greater is the value of $Rel(n_c, n_j)$, the higher is the probability of n_j to become the next step of the crawler. The nodes are interlinked in $\mathcal{G}_s$ based on their relevance factor with each other in the graph. For instance, a current node n_c is linked to its neighbors $\mathcal{N}\left(n_c\right)$, if $\mathcal{N}\left(n_c\right)$ fulfills the applied relevance constraints on inclusion to $\mathcal{G}_s$ as $f(x) \subseteq \mathcal{G}_s$ and $f(x)$ is defined as:

$$f(x)_{\forall\Gamma|\Gamma\in\mathcal{N}\left(n_c\right)} = \begin{cases} 1, & \mathit{if}\ \mathcal{G}\left(n_c\right) \cdot \mathcal{I}_\ell \geq \vartheta \\ 0, & \mathit{if}\ \mathcal{G}\left(n_c\right) \cdot \mathcal{I}_\ell < \vartheta \end{cases} \tag{11}$$

Where 1 shows the existence of influential connection between neighbors, 0 represents no connection, ϑ is a threshold and its value is 30% in this case, and $\mathcal{G}\left(n_c\right)$ defines generalization expression for *Rel* as

Algorithm 1. NRG-based Meta-Path Selection and Influential Graph Construction

```
     Inputs:   nodes, paths, h = 0, H = 5: Hop index and Hop length respectively, $G$: KG, $\mathcal{I}_\ell$: Interaction log,
               $\mathcal{M}$: Interaction Matrix, $\mathcal{N}$: Neighbors, $d$: embeddings.
     Output:   Similarity Attributed Meta-Path-based Influential Graph.
1    foreach $(n_c, n_j)$ pair do
2        Reset h
3        while (h! = H) do
4            $n_j \leftarrow$ nearest $\mathcal{N}(n_c)$    // the nearest neighbor of $n_c$ is termed as $n_j$
5            if(cmp$(n_c, n_j) > \vartheta$ according to $Relevance(n_c, n_j)$ in Eq. (10)
6                call Shade            // procedure is called
7                else
8                $n_j \leftarrow$ Next nearest $\mathcal{N}(n_c)$ & goto line. 5
9            end if
10           if $|\mathcal{N}(n_j)| > 0$
11               $n_c \leftarrow n_j$ & inc h     // for hop-2, repeat the process
12               else                     // if $n_j$ has no neighbors, consider next neighbor of $n_c$ as $n_j$
13               $n_j \leftarrow$ Next nearest $\mathcal{N}(n_c)$ & goto line. 5
14           end if
15       end while
16       identify & store $\wp$ as $\wp[\,] \leftarrow \wp_i$ //$i = 1$, on completion of one iteration of h
17   end foreach
18   concat $\wp$ from $\wp[\,]$wrt id          //meta-paths IDs-based interlinkage
19   proc shade
20       $n_j$: if not highlighted, highlight
21       $n_j$: add to $\mathcal{G}_s$
22       concat $(n_c, n_j)$
23   end proc
```

$$\mathcal{G}(n_c) = \wedge_{i=1}^{s}\{n_{c-i}|Rel(n_c, n_{c-i}) > Rel(n_c, n_{c-J})\}n_{c-i}, n_{c-j} \quad (12)$$

Where $\mathcal{G}(n_c)$ is the first-order influential graph, $\wedge$ is the sequential concatenation operator, n_{c-i} is the next node that is linked to n_c, and n_{c-j} is the node that is not-connected to n_c.

4.2 Hashing module

In this section, we transform $\mathcal{G}_s$ to hash-codes, place identical hash-codes in identical hash-buckets, retrieve information and generate the final outcomes.

4.2.1 Transformation

We transform k-dimensional actual-valued continuous-embeddings n_k to b-dimensional binarized embeddings n_b, where $|k| = |b|$. We applied *tanh* activation function for transformation as

$$f_b = \tanh(W^{\top} n_k + z) \quad (13)$$

Where $W \in \mathbb{R}^{k\times b}$ and $z \in \mathbb{R}^b$ show weight parameters and bias factor respectively. We independently denote u_i-v_j hash-codes as $h_{ui} = h_{uj} = \{\pm 1\}^{\mu}$, where μ represents the length of bits. For formal hash-codes h_i, we applied *sign* function to formalize f_b wrt all instances as, if $f_{bi} \geq 0$ the function enters $+1$ against each i into $\mathcal{M}$ otherwise-1. To overcome the limitations[5] of the *sign* function, we incorporated SRS[6] [41] with margin weight γ on continuous increase, as an activation function, till the convergence of instances. We defined the SRS through margin weight γ as follows.

$$SRS(\gamma, x) = \frac{\gamma \cdot x}{\frac{\gamma\cdot x}{\alpha} + e^{\frac{-\gamma\cdot x}{\beta}}} \quad (14)$$

Where $\gamma \mid \gamma > 0$ shows the cumulative margin weight and x is the actual input. Formally, with a continuous increase in the value of γ, SRS approaches the original sign function, as

$$\lim_{\gamma\to\infty} SRS(\gamma, x) = \text{sign}(x) \quad (15)$$

Thus, the hash-codes of the corresponding entities are preserved, but to enhance the performance via maximizing the Hash-Hits, it is essential to embed the similar hash-codes into the similar hash-buckets.

4.2.2 Identical hash-codes to identical hash-buckets

Although the orthogonal arrangement of instances is feasible, we verified their apparent relevance with their corresponding neighbor's through Deep-Probabilistic (*dProb*) technique, as shown in Algorithm 2.

We provided the hash-codes h_i to the DNN as $h_i = [i = 1, 2, \ldots, K]$ to calculate the mutual likelihood of the corresponding entities through the following process:

$$h_i^{(k+1)} = \sigma\left(W^{(k)} \cdot h_i^{(k)} + b^{(k)}\right)\ s.t.\ \mathcal{L} = \lambda_k \|\Theta\|_2^2 \quad (16)$$

Where k is the depth of layer and σ is the non-linear activation function; $W^{(k)}$ represents the model's weight matrix wrt k, $h_i^{(k)}$ shows the outcome and $b^{(k)}$ is bias factor of kth layer; $\mathcal{L}$ is the loss function, λ_k is the regularization of kth layer, Θ describes the set of model's hyper parameters, and $\|\cdot\|_2^2$ is the Euclidian norm.

Through the *dProb*, we formulated the interlinking probability N_{cj} of n_c and n_j wrt their concerned hash-codes h_c and h_j respectively. N_{cj} is defined as:

$$\mathcal{P}(\mathrm{N}_{cj}|h_c, h_j) = \Phi^{\mathrm{N}_{cj}} \cdot \Psi^{1-\mathrm{N}_{cj}} = \begin{cases} \mathrm{N}_{cj} = 1, & if\ \Phi \\ \mathrm{N}_{cj} = 0, & if\ \Psi \end{cases} \quad (17)$$

Additively, path p_z selection probability P_{cz} by n_c to go to n_j, to consider it as relevant node, wrt to their concerned hash-codes h_c and h_z respectively. P_{cz} is defined as:

$$\mathcal{P}(\mathrm{P}_{cz}|h_c, h_z) = \Phi^{\mathrm{P}_{cz}} \cdot \Psi^{1-\mathrm{P}_{cz}} = \begin{cases} \mathrm{P}_{cz} = 1, & if\ \Phi \\ \mathrm{P}_{cz} = 0, & if\ \Psi \end{cases} \quad (18)$$

Thus, the total probability is the sum of Eq. (17) and (18), we have:

$$\mathcal{P}(\mathrm{N}_{cj}|h_i \mid i = 1, 2, \ldots, K) = \frac{\mathcal{P}(\mathrm{N}_{cj}|h_c, h_j) + \mathcal{P}(\mathrm{P}_{cz}|h_c, h_z)}{2} \quad (19)$$

where $\Phi = x\cdot\sigma(x)$, $\Psi = 1 - x\cdot\sigma(x)$; $x = \Delta(h_c, h_j)$, σ is sigmoid function, i.e., $\sigma(x) = (1 + exp(-x))^{-1}$, Δ represents the distance; and to activate the hash-codes, we used the swish activation function, i.e., $f(x) = x\cdot\sigma(x)$ [42].

The $\mathcal{P}(\mathrm{N}_{cj}|h_i \mid i = 1, 2, \ldots, K)$ is subjected to the probabilistic density function $f_D(\Pi)$ as $\mathcal{P}(\mathrm{N}_{cj}|h_i \mid i = 1, 2, \ldots, K) \in f_D(\Pi)$ to preserve the selection head in the data-range determined by the density function $f_D(\Pi)$, defined as:

$$f_D(\Pi) = \mathcal{P}(x_{min} \leq \pi \leq x_{max}) = \int_{x_{min}}^{x_{max}} f(\pi)d\pi \geq 0 \quad (20)$$

The information instances are placed in relevant hash-buckets-based on their mutual likelihood, as shown in the Fig. 2, to enhance the hit-ratio of hash functions up to a maximum possible extent.

[5] Zero gradient on non-zero inputs, non-zero mean, false or negative missing, unrealistic response on zero-input

[6] Soft-Root-Sign activation function

Algorithm 2. Similar Hash-Codes to Similar Hash-Buckets

```
Inputs:  $e$: Entities Information Instances, $r$: Relation Information, $h$: Corresponding Hash-
         Codes, $\mathcal{M}$: Interaction Matrix. $\Phi = x \cdot \sigma(x)$, $\Psi = 1 - x \cdot \sigma(x)$; $x = \Delta(h_c, h_j)$,
         $\sigma(x) = (1 + exp(-x))^{-1}$
Output:  Suggestions of Hash-Buckets to Specific Hash-Codes
1   Pre-train embedding
2   do
3       foreach $(\mathrm{n_c}, \mathrm{n_j})$ in $\mathcal{M}$ do
4           func N2N-P
5               for i = 1 to K wrt $\mathrm{H}(\mathrm{h_c}, \mathrm{h_j})$ do
6                   $\mathcal{P}(\mathrm{N}_{cj}|h_c, h_j) = \Phi^{\mathrm{N}_{cj}} \cdot \Psi^{1-\mathrm{N}_{cj}} = \begin{cases} \mathrm{N}_{cj} = 1, & if\ \Phi \\ \mathrm{N}_{cj} = 0, & if\ \Psi \end{cases}$
7                   return $\mathcal{P}(\mathrm{N}_{cj}|h_c, h_j)$
8               end for
9           end func
10      end foreach
11      foreach $(\mathrm{n_c}, \mathrm{p_z})$ in $\mathcal{M}$ do
12          func N2P-P
13              for i = 1 to K wrt $\mathrm{H}(\mathrm{h_c}, \mathrm{h_j})$ do
14                  $\mathcal{P}(\mathrm{P}_{cz}|h_c, h_z) = \Phi^{\mathrm{P}_{cz}} \cdot \Psi^{1-\mathrm{P}_{cz}} = \begin{cases} \mathrm{P}_{cz} = 1, & if\ \Phi \\ \mathrm{P}_{cz} = 0, & if\ \Psi \end{cases}$
15                  return $\mathcal{P}(\mathrm{P}_{cz}|h_c, h_z)$
16              end for
17          end func
18      end foreach
19  while (Mod)! Converge
20  do
21      foreach $(n_c)$ in returned-data do
22          for i = 1 to K do
23              call N2N-P
24              call N2P-P
25              $\mathcal{P}(\mathrm{N}_{cj}|h_i \mid i = 1, 2, \ldots, K) = \frac{\mathcal{P}(\mathrm{N}_{cj}|h_c, h_j) + \mathcal{P}(\mathrm{P}_{cz}|h_c, h_z)}{2}$
26              store $\mathcal{P}(\mathrm{N}_{cj}|h_i)$ in bucket $i$
27          end for
28      end foreach
29  while (main)! Converge
```

4.2.3 Information retrieval

Data retrieval via hashing-method recovers the loss of information with radial-based data spreading technique. It captures the local-structure of information and reduces its feature's dimensionality via nonlinear-projection. The retrieval process is presented in Algorithm 3. Moreover, graph anchors are incorporated to highlight the selective features (hash-codes) in the information instances. We configured the graph anchors to alleviate the burdensome of blind matching in the buckets. We transformed the items of A to the representations of graph anchor as:

$$f(a) = e^{0.5\times\frac{\|h_i-a\|_2^2}{\sigma^2}} \tag{21}$$

Where $f(a)$ is a function used to represent the graph anchors and $i = 1, 2, \ldots, \upsilon$, and $\|h_i - a\|_2^2$ shows the Euclidean distance between the selected feature i and the graph anchor a.

The defined linear models E, R, H_e, H_r are binary representations of information in hamming space, therefore according to [36], linear auto-encoder regression (LAR) is feasible for projection. To persist the semantic relevance among the information instances, we applied LAR for matrix projection to minimize the regression loss and used transpose for matrix multiplication to optimize the computational overhead. We performed two-way linear projection from H_e into E (1st way) and conversely with the graph anchored factor a from E into H_e (2nd way). We aggregated the projections via additive association by keeping the hit-rate maximized among the instances of H_e and E matrices. The objective function is formulated as:

$$\max_{H_e,E}\left\|E - \left(\left(H_e^{\top}\right)^{-1}E^{\top}\right)^{\top}H_e\right\|_2^2 + \xi\left\|H_e - H_eE^{\top}\left(EE^{\top}\right)^{-1}E\right\|_2^2 \tag{22}$$

Algorithm 3. The Hashed Retrieval Process

```
Inputs:  $h$: Hash-Buckets, H: Hash-codes, $f_c$: Hash-function call, $f(\alpha)$: Anchor function, P: Hash-
         Projections, $\Upsilon = \mathcal{N}(i)^{sim_{\alpha,\beta} \times (\hat{R}_{\alpha,\beta} - \delta_i)}$ and $\mathrm{T} = \mathcal{N}(i)^{|sim_{\alpha,i}|}$.
Output:  Required Hash-Codes
1        Projections P to the Hamming-space;
2        do
3            foreach index $\alpha$ in P do
4                func H-Retrieval
5                    for $i = 1$ to K wrt $\mathrm{H}_i$ do
6                        $\beta = \frac{\mathcal{P}(\mathrm{N}_{cj}|h_c, h_j) + \mathcal{P}(\mathrm{P}_{cz}|h_c, h_z)}{2}$
7                        If (attack $h_i$ via $f_c(\beta)$) give 1
8                            $\bar{R}_{\alpha,\beta} = H(\beta)$
9                            return $\bar{R}_{\alpha,\beta}$
10                           else
11                           $\hat{R}_{\alpha,\beta} = \frac{R_u + \sum_{i=1}^{K} \in H(\alpha,K) \cap \Upsilon}{\sum_{i=1}^{K} \in H(\alpha,K) \cap \mathrm{T}}$
12                           return $\hat{R}_{\alpha,\beta}$
13                       endif
14                   end for
15               end func
16           end foreach
17       while (Mod)! converge
18       do
19           foreach ($h_i$) in returned-data do
20               for i = 1 to K do
21                   call H-Retrieval
22                   assign unique ID to the retrieved $h_R$
23                   store $h_R$
24               end for
25           end foreach
26       while (main)! converge
```

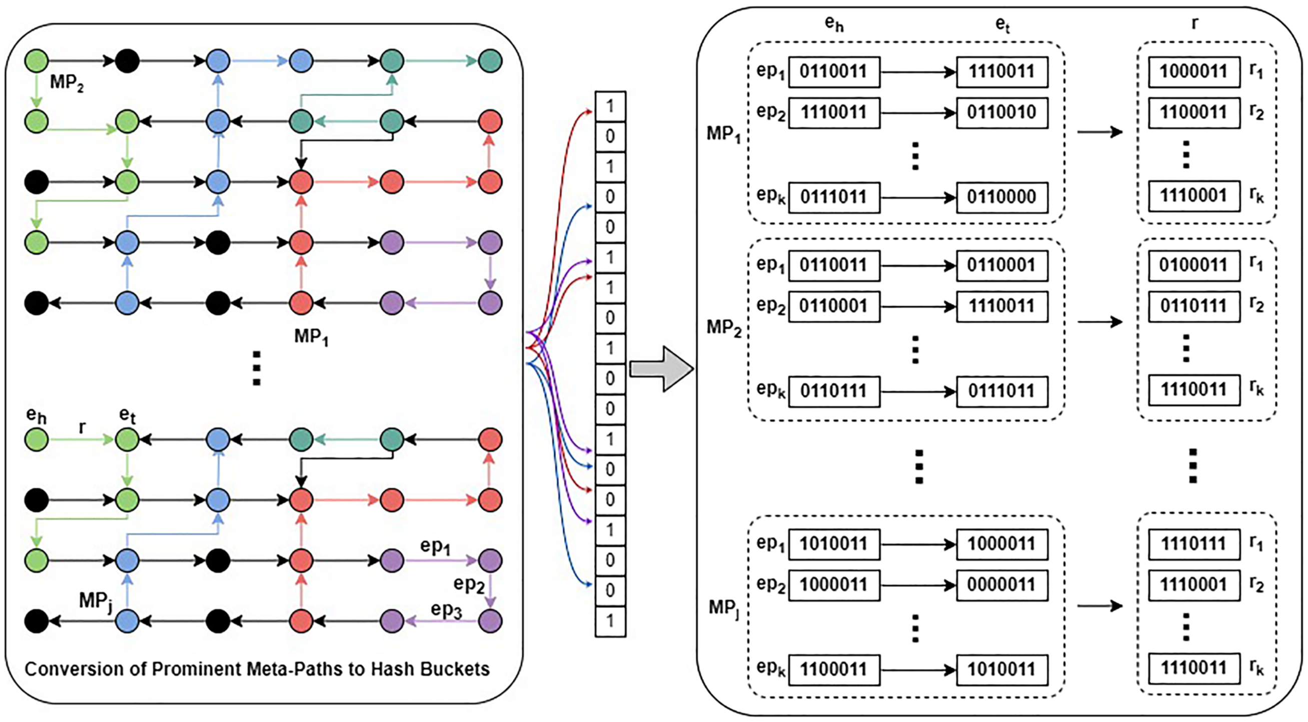


Fig. 2 Conversion of prominent Meta-Paths to Hashed Tracks of Binarized Information. Abbreviations used: MP – Meta Path, ep – Entity Path (one relation distance between two nodes), r – relation, e_h & e_t – Head & Tail entities respectively

Similarly, we projected H_r and R into each other by defining through the following expression:

$$\max_{H_r,R}\left\| R-\left(\left(H_r^{\top}\right)^{-1}R^{\top}\right)^{\top}H_r\right\|_2^2+\xi\left\|H_r-H_rR^{\top}\left(RR^{\top}\right)^{-1}R\right\|_2^2 \tag{23}$$

Where ξ shows a hyper parameter used to counterbalance the projection impact on both sides. After the required projections, we again applied *dProb* via Eq. (19), to attack the hash-buckets via function calls $H(\alpha)$ to directly retrieve the required hash-codes and return them to the presentation module.

$$\overline{R}_{\alpha,\beta}=H(\beta) \tag{24}$$

Where $\overline{R}$ represents the sequential retrieval, α shows the index of f_c to hit the target, H represents the hash-bucket and β is the specific hash-code being retrieved during the Hash-Hit in Eq. (19). In case of Hash-Miss, we call LS function exactly from the location in the bucket that returned 0 to the 1st call, and try to retrieve the required code in a margin of 3 indices above and below the location of Hash-Miss. The LS expression for retrieval is described as:

$$\hat{R}_{\alpha,\beta}=\frac{R_u+\sum_{i=1}^{K}\in H_i(\alpha,K)\cap \mathrm{Y}}{\sum_{i=1}^{K}\in H_i(\alpha,K)\cap \mathrm{T}} \tag{25}$$

Where $\hat{R}_{\alpha,\beta}$ is Relevance between α and β, R_u is mean of all user references, H_i is the corresponding hash-buckets of the very specific hash-codes, $\mathrm{Y}=\mathcal{N}(i)^{sim_{\alpha,\beta}\times(\hat{R}_{\alpha,\beta}-\delta_i)}$ and $\mathrm{T}=\mathcal{N}(i)^{|sim_{\alpha,i}|}$.

Any set of bits $\Delta\in\{-1,1\}^{\mu}\mid\Delta\subset HB\wedge\{-1,\ 1\}^{\mu}\in\mathcal{E}$ represents a matrix of binary instances. In $\{-1,\ 1\}^{\mu}$, where $\mu=n_b\times k$, each tuple is a hash-code of instance n_i of n_b dimension and k is the total number of instances. In order to preserve the semantic relevance among the representations of identical instances and to optimize their entropy, we exploit un-correlation of each bit as $\Delta\Delta^{\top}=\mathrm{I}_n$ and balance of each bit as $\Delta \mathrm{b}_k=0$, where $\Delta\Delta^{\top}$ is a dot product to acquire the unity diagonal index I_n of vector of b_k with a length k having all ones as $b=1$. Typically, to make the objective function to retrieve the identical hash-codes in the maximum number of its iterations, un-correlation demands to preserve the condition that each Δ-row must not be co-related with any other row-index of Δ by ignoring the concern of correlation of the column-index; whereas, balance demands to preserve the condition that each bit must demonstrate 1 in half of the frequency of its occurrences. We applied these constraints through the extended normalization technique, i.e., Elastic Net Regularization (ENR) [43].

4.3 Presentation module

This module triggered the hash calls; and in return collected, IDfied and stored the responses wrt the function calls in the result pool. Thus, the received hash-codes about the inter-entity interactions are processed and potential preferences are generated as;

$$\hat{\delta}_{(u,v)}=\varphi\left[\left(h_u,h_v\right)\otimes\left(\overline{R}_{\alpha,\beta},\hat{R}_{\alpha,\beta}\right)\right] \tag{26}$$

Where $\hat{\delta}_{(u,v)}$ is the required preference score from user u to the potential item v, φ is multilayer perceptron, $\bigotimes$ is focus projection operator and, h_u and h_v are the hash-codes of the concerned information instances of users and items (i.e., the source and destination h-codes), and $\overline{R}_{\alpha,\beta}$ and $\hat{R}_{\alpha,\beta}$ are hash hit and miss expressions, respectively. We adopted an equally divide and conquer strategy on each higher layer by overwhelming the hidden units to model the abstracted information. We applied Swish to activate the hidden layers and to regulate the outcomes of $\hat{\delta}_{(u,v)}$ between 0 and 1 wrt the Elastic Net Regularization. Finally, based on the predicted preferences about the potential interests of the users, this module generated the recommendation results.

4.4 Optimization

ENR is aggregation of ridge and lasso coefficients, i.e., ℓ_1 and ℓ_2 norm regularizations respectively, to generate an optimized output. ENR creates λ elastic net by tuning α parameters to 0 for ridge and 1 for lasso coefficients. In ENR, we selected α between 0 and 1 to optimize the elastic-net[7] to effectively shrink the coefficients and to set them to 0 for dealing with the sparse selections [44]. In overall regularization, we applied ENR to optimize the experimental setup via the following loss function.

$$J(\beta_1,\beta_1,\ldots,\beta_m)=\sum_{i=1}^{n}\left(y_i-\sum_{j=1}^{m}x_{ij}\beta_j\right)^2+\lambda\left(\alpha\sum_{j=1}^{m}|\beta|+\frac{1-\alpha}{2}\sum_{j=1}^{m}\beta_j^2\right) \tag{27}$$

Where β represents the retrieved instances, x and y are the classes of present and required instances, α is the coefficient of ENR-regularization set for f_c, and λ is a hyper-parameter to counter-balance the impact of loss.

4.4.1 Testimony loss

Transformation outputs fall in two categories wrt completion, i.e., correctly transformed and reported with 1, and faced some issue and reported with 0. The correctly transformed instances belong to pool $|^{(k)}$, and rest are considered as the transformation forfeiture (loss), that is defined as:

$$\mathcal{F}_{TF}=\sum_{(t_i)\notin|^{(k)}}\log\left[1-f\left(x_t\cdot\sigma\left(x_t\right)\right)\right]-\sum_{(t_i)\in|^{(k)}}\log\left[f\left(x_t\cdot\sigma\left(x_t\right)\right)\right]+\lambda\|\Omega\|_2^2 \tag{28}$$

[7] The term is "elastic" because weightage of any or all regularization-parameters can be adjusted according to the requirements of the experimental setup.

Where $\mathcal{F}_{TF}$ is the testimony of transformation forfeiture, t is the index of transformation, $f(x) = \mathrm{W}|^{(k)} + b$, λ is ENR coefficient, Ω is the set of model's parameters and $\|\cdot\|_2^2$ is Euclidian norm.

4.4.2 Interaction loss

In information retrieval, the observed interactions between user and item are considered as 1 and not-observed (or missing) interactions as 0. The 1 s possess upper edge in loss declaration as compared to 0 s because they contribute to preference generation [45]. The testimony of optimal interaction loss is expressed as:

$$\mathcal{F}_{IL} = \sum_{(u,j)\in\mathcal{I}^0} \log\left[1 - \left(\hat{\delta}_{(u,j)} \cdot \sigma\left(\hat{\delta}_{(u,j)}\right)\right)\right] - \sum_{(u,v)\in\mathcal{I}^1} \log\left[\hat{\delta}_{(u,v)} \cdot \sigma\left(\hat{\delta}_{(u,v)}\right)\right] \quad (29)$$

Where $\mathcal{I}^1$ and $\mathcal{I}^0$ are the observed and un-observed interactions among users and items, respectively.

4.4.3 Training

According to [46], we optimized the overall loss via SGD[8] that streamlines learning-rate according to the absolute-value of the concerned gradients. We trained hyper-parameters via back propagation by maximizing the log-likelihood of $(u, v) \in \mathcal{I}^1$ as:

$$\mathrm{Y}_{train} = \sum_{s=1}^{S} \sum_{(u,v)\in\mathcal{I}^1} \log \Pi\left(\hat{\delta}_{(u,v)} \cdot \sigma\left(\hat{\delta}_{(u,v)}\right)\right) \quad (30)$$

Where Π operates the swish activator that is triggered via softmax-function as:

$$\Pi\left(\hat{\delta}_{(u,v)} \cdot \sigma\left(\hat{\delta}_{(u,v)}\right)\right) = \frac{\exp\left(\hat{\delta}_{(u,v)} \cdot \sigma\left(\hat{\delta}_{(u,v)}\right)\right)}{\sum_{\forall(u,V)\in\mathcal{I}^1} \exp\left(\hat{\delta}_{(u,V)} \cdot \sigma\left(\hat{\delta}_{(u,V)}\right)\right)} \quad (31)$$

4.4.4 The overall loss (forfeiture)

Collectively, the overall loss belongs to four different aspects of the proposed approach; i.e., (i) Relevance calculation $\mathcal{F}_{Rel}$, (ii) Translation $\mathcal{F}_{TL}$, (iii) Transformation $\mathcal{F}_{TF}$, and (iv) Decoding $\mathcal{F}_{Dec}$. For (i), we declared relevance error-ratio between the relevant and the irrelevant nodes as $\mathcal{F}_{Rel} = \sum_{\forall Y\notin\mathcal{G}} \log\left[1 - (Rel(Y) \cdot \sigma Rel(Y))\right] - \sum_{\forall Z\in\mathcal{G}} \log\left[Rel(Z) \cdot \sigma Rel(Z)\right]$ Where $Y=(a,b)$ are the nodes that are mutually irrelevant – suggested by the function, and $Z = \left(n_t, n_c\right)$ are relevant ones. For (ii), we declared $\mathcal{F}_{TL} = \sum_{\forall B\notin\bar{G}} \log\left[1 - (g(B) \cdot \sigma g(B))\right] - \sum_{\forall A\in\bar{G}} \log\left[g(A) \cdot \sigma g(A)\right]$ where $B|B=(a,p,b)$ describes a triplet that is invalid because the facts in a or b or both are either unreadable or missing. Although the required information is automatically inserted to the concerned representations by the regularization-layer of the translator based on the structure of triplets, yet the triplets are not-validated by the triplet-validator [47], whereas $A = \left(n_t, p, n_c\right)$ is a validated triplet. For (iii), $\mathcal{F}_{TF}$ is declared in Eq. (28), and similarly for (iv), $\mathcal{F}_{IL}$ is given in Eq. (29). Therefore, $\mathcal{F}_{H-SAGE} = \mathcal{F}_{Rel} + \mathcal{F}_{TL} + \mathcal{F}_{TF} + \eta\mathcal{F}_{IL}$ is the total loss of the proposed approach; where η is a hyper parameter used to counter-balance the impact of overall loss.

4.4.5 The time complexity

Collectively, the Time Complexity (TC) belongs to five different aspects of H-SAGE; (i) *Rel*, (ii) Meta-Path aggregation, (iii) Translation and Transformation, (iv) Hashing on Hit, and (v) Hashing on Miss. For (i), the TC of entity-relevance is $O(|Rel|d^2)$ and that of semantic-relevance is $O(|SRel|d)$ where, *SRel* is semantic-Relevance. So, the total TC of (i) is $O(|Rel|d^2 + |SRel|d) \simeq O(|Rel|d^2)$. For (ii), we collected the facts wrt aggregation of meta-paths from (i), that possesses the TC of modeling entities to meta-paths and meta-paths to influential graph. Therefore, $O(|pN|d + |plogN|d)$ and $O(|\wp M|d + |\wp logM|d)$ are TCs of aggregation of paths and meta-paths respectively. So, the total TC of (ii) is $O(|pN + \wp M|d + |plogN + \wp logM|d) \simeq O(|C|d + |logC|d)$, where p is path, N is number of paths in meta-path, $\wp$ is meta-path, M is number of meta-paths in the influential graph and C is constant. Similarly, for (iii), the TC for translation is $O\left(\left|\bar{G}\right|d^2\right)$ because it belongs to the construction module in offline processing, we already discussed it in (i); and in case of transformation, it possesses offline linear computing time of $O\left(\sum_{i=1}^{k} \left|\bar{G}\right|d_i\right)$, where $\bar{G}$ is the influential graph. For (iv), TC is $O(1)$, and for (v), TC is $O\left(\sum_{i=1}^{3\times3} |\mathrm{H}|b_i\right)$, where H is hash-code and b is bucket. By adding the TCs of all parts, we have $|M| \sum_{s=1}^{S} \left|\bar{G}_s\right|d_s^2 \simeq O\left(n^2\right)$ as the highest optimal TC of non-hashing part of H-SAGE; that is not always executed. On the other hand, TC of Hash-Hit and Hash-Miss is $O(1)$ and $O\left(\sum_{i=1}^{3\times3} |\mathrm{H}|b_i\right) \simeq O(n)$ respectively. Thus, the overall TC of H-SAGE is feasible for experiments on preprocessed datasets.

Concerning to the individual TC of the proposed algorithms, we provide statement-wise computational time $t(n)$ and total computational time $T(n)$ of each algorithm in the following section.

Algorithm 1 In this algorithm, we fetch prominent meta-paths in the raw data and inter-connect them to construct the subgraph. We mention the $t(n)$ as: foreach $(\mathrm{n_c}, \mathrm{n_j})$ pair do; $t(n) = n$. The TC of all statements in the outer-loop is $t(n) = n$, but these statements are not involved in the sequential computation, therefore, their $t(n) = 1$. In the inner block; while $(\mathrm{h} \,!= \mathrm{H})$ do; $t(n) = n$, $\mathrm{n_j} \leftarrow$ nearest $\mathcal{N}(\mathrm{n_c})$; $t(n) = n + 1$, if(cmp$(\mathrm{n_c}, \mathrm{n_j}) > \vartheta$; $t(n) = 2n - 1$, if $\left|\mathcal{N}(\mathrm{n_j})\right| > 0$; $t(n) = \log n$, identify & store $\wp$ as $\wp[\,] \leftarrow \wp_{\mathrm{i}}$;

[8] Stochastic Gradient Descent

$t(n)=n$, Rest, in all of the k statements; $t(n)=n=kn$. The TC of the inner block is $T(n)=n+n+1+2n-1+\log n+kn=5n+kn+\log n$, and the total TC via the outer loop is $T(n)=n(5n+kn+\log n)=5n^2+kn^2+nlogn\cong O(n^2)$.

Algorithm 2 This algorithm is based on YES/NO strategy in a sense the required hash-code FOUND/NOT-FOUND to move that to the hash-buckets. In other words, the inner calculations rely on discrete values having no concern with the sequential processing and management. The $t(n)$ is as: do; $t(n)=n$; is initiator, foreach $(\mathrm{n_c}, \mathrm{n_j})$ in $\mathcal{M}$ do; $t(n)=n$, func N2N-P; $t(n)=n$, for i = 1 to K wrt $\mathrm{H}(\mathrm{h_c}, \mathrm{h_j})$ do; $t(n)=n$, $\mathcal{P}(\mathrm{N}_{cj}|h_c, h_j) = \Phi^{\mathrm{N}_{cj}} \cdot \Psi^{1-\mathrm{N}_{cj}}$ in Eq. (17); $t(n)=\log n$, $\mathcal{P}(\mathrm{P}_{cz}|h_c, h_z) = \Phi^{\mathrm{P}_{cz}} \cdot \Psi^{1-\mathrm{P}_{cz}}$ from Eq. (18); $t(n)=\log n$, $\mathcal{P}(\mathrm{N}_{cj}|h_i \mid i = 1, 2, \dots, K)$ in Eq. (19); $t(n)=constant$, return $\mathcal{P}(\mathrm{N}_{cj}|h_c, h_j)$; $t(n)=n$, store $\mathcal{P}(\mathrm{N}_{cj}|h_i)$ in bucket i; $t(n)=n$, For rest of the k statements; $t(n)=kn$. Hence expressively, the total TC is $T(n)=n(n(klogn)+kn)=n^2klogn+kn^2\cong O(n^2)$; but this algorithm have no sequential processing. For each outer-loop, the instant-inner loop is just iteration if there is no sequential processing, deem the computation in such nested loops with additive property as $T(n)=O(n+n)$ instead of multiplicative property as $O(n*n)$. Thus, in our case the overall $T(n)$ is $n+n+klogn+kn\cong O(n)$.

Algorithm 3 In this algorithm, we retrieve the target hash-code via Hash-hit or Hash-Miss. The $t(n)$ is: do; $t(n)=n$; it is basically the initiator, foreach index α in P do; $t(n)=n$, func H-Retrieval; $t(n)=n$, for $i=1$ to K wrt H_i do; $t(n)=n$, β in Eq. (19); $t(n)=constant$; if it gives 1 (Hash-Hit), β; $t(n)=\log n$; if it gives 0 (Hash-Miss), In case of 0, we need to do the following process: $\hat{R}_{\alpha\beta}$ in Eq (25); $t(n)=\log n$, For rest of the k statements; $t(n)=kn$. Hence based on the discussion in Algorithm 2, in case of Hash-Hit, $T(n)=n(constant)\cong O(1)$, because β gives 1, and an indication along-with the target hash-code is sent to the main function saying that this hash-code has matched and control is also transferred to the main function, therefore, the TC of Hash-Hit is $O(1)$. On the other hand, during the Hash-Miss, β gives 0, and the statement after the else statement is executed for locality-sensitive hashing. Therefore, the total TC in Hash-Miss is $T(n)=nlogn\cong O(n)$.

5 Experiments

In this section, we deal with the following research questions – the main objectives of this work.

RQ1: Can H-SAGE outperform the-state-of-the-art methods wrt performance?
RQ2: Can H-SAGE outperform the-state-of-the-art methods wrt computational complexity?
RQ3: How is the performance of H-SAGE in dealing with the issues of data-sparsity?
RQ4: What is the impact of different modules of H-SAGE on performance?
RQ5: Is H-SAGE sensitive to the different adjustments of hyper-parameters?
RQ6: Can H-SAGE provide explainable personalized recommendations?

5.1 Experimental setup

In this section, we discuss the exploited datasets and the applied evaluation techniques (metrics). Also, we define the baseline methodologies selected for comparison with H-SAGE and the environment settings of hyper-parameters.

5.1.1 Data and data pre-processing

We came across extensive experimental work on three real world benchmark datasets, i.e., Amazon-Book, Last-FM and Bing-News to evaluate the performance of H-SAGE. **Amazon-Book**[9] contains users, items, interactions and over 20 M user ratings (in a range of 1 to 5) about the books. It is retrieved from Amazon-product's data – a widely used knowledge base for various product's recommendation [48]. **Last-FM**[10] contains information about the performance of musicians and previous music-listening records of the interacted users (i.e., music-tracks). The tracks are considered as items. The information is retrieved from online music-system, i.e., Last.fm [49]. **Bing-News,**[11] also known as MIND[12] - an assemblage of implicitly collected user-feedbacks from the server-side logs of MS News,[13] contains users with their feedbacks and small statements of news with titles from December 22, 2020, to May 30, 2021, according to [50]. There exist many other benchmark and ordinary datasets having various versions or variations like Movie-Lens (e.g., Movie-Lens-100 K, Movie-Lens-1 M, Movie-Lens-10 M, etc.), YELP (e.g., Yelp-2013, 2014, 2018, etc.), Douban-Book, Book Crossing, KKBox, CEM, Dianping-Food, etc. But, by keeping the space limitations in our consideration, we selected only these three benchmark datasets to perform the experiments because they are populated and hereby commonly utilized datasets. In particular,

[9] http://jmcauley.ucsd.edu/data/amazon/
[10] http://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-readme.txt
[11] https://www.bing.com/news
[12] MIcrosoft News Dataset, https://msnews.github.io/
[13] https://microsoftnews.msn.com/

they contain explicit/implicit ratings from the users towards items in different ranges (i.e., 1–5/1–10) and provide positive argumentation to augment the decision-making process.

In the datasets, although we have users-to-items interactions, we are required to create an enriched item knowledge-base for each dataset with or without the help of an external knowledge-base – to enrich the entity representations of the datasets. Therefore, for Amazon-Book, we preprocessed the actual dataset to highlight the interactions between users and items. As initial input, we used ID-embeddings of entities and considered 1 if we obtained any interaction between users and books. Further according to [50], we identified triplets with word "book" anywhere in the context of the dataset, validated via the imposed constraint of RAV[14] >0.5 among the nodes, and retrieved their entities and relations to enrich the entities of the local subgraph. To maintain the consecutive streams of relevant information in the graph, we enriched Meta paths by incorporating such triplets in the concerned sequence that had "book" in minimum at 3 places out of 5 in $\wp$. We guaranteed feasible relations among entities in the $\wp$ via unique triplet identifiers, i.e., TIDs, to preserve the synchronized relations among old and new information instances in the Meta-paths. To further enrich the underlying information, we kept the RAV fixed and used a formatted query (i.e., $<e_h, r, e_t> = <*.book.*, *.book.*, *.book.*>$) to retrieve feasible and relevant information from MS-Satori. Finally, we used TransE [47] to validate triplet's granularity and redundancy and discarded the inappropriate instances.

According to KB4Rec,[15] we retrieved map-able information instances and their mappings from DBpedia-ontology[16] and exploited to enrich the entity-information of the local subgraph of Last-FM. We aligned the data-items of Last-FM according to the highlighted information instances of the external KG, and retrieved only those instances that have a relation-frequency $r_f \geq 3$ wrt their first-order neighbors. We continued the process under the applied constraints to enrich each data-item in the local subgraph up to the Meta-path extent of 5 hops from every current item. Moreover, for Bing-News we mapped the embeddings of corresponding items of the subgraph, via concatenation of item-IDs, to the embeddings of string-tokens of titles and statements of news retrieved from MS-Satori. The information in string-tokens of news-titles and statements is greater than information in titles of books and music-tracks; thus, Bing-News is comparatively more feasible for effective decision making.

Table 1 The Statistics of Datasets Utilized

Literals	Datasets		
	Amazon-Book	Last-FM	Bing-News
Domain	Books	Music	News
Users u	55,255	1865	40,237
Items v	20,235	6526	32,562
Interactions $\mathcal{I}$	232,562	68,456	192,356
Entities	77,253	12,039	76,586
r-Types	29	58	45
r-Counts	122,562	29,650	115,620
Sparsity sp	0.999792	0.994375	0.999853
Training Y_{train}	162,793	47,919	134,649
Testing Y_{test}	46,512	13,691	38,471
Validation Y_{val}	23,256	6845	19,235

We removed those items that had no relevant mappings in the external KGs as well as users with ratings <5, and calculated the data sparsity via *"subtracting the ratio of interactions to the product of users and items from 1"* as $sp = 1 - \frac{\mathcal{I}}{u \times v}$. Moreover, we divided the datasets in 70%, 20% and 10% of ratings as Training, Testing and Validation, respectively. The statistics are displayed in Table 1, as well as we released and published the datasets in Mendeley-Data entitled *"H-SAGE-Dataset"*.[17]

5.1.2 Evaluation

We used well-known *CTRP*[18] and *top-K recommendation* for performance evaluation. In CTRP, we calculated the predictive possibility of next click wrt the testing data given the learning interactions of training data. We applied (AUC & Acc)[19] for the performance evaluation of H-SAGE and baseline methods wrt CTRP. Similarly, we obtained top-K items with having the highest probability of the next possible click via utilization of training data-interactions for each instance of user in the testing data. We applied (Prec, Rec, NDCG)@K[20] for the performance evaluation of top-K recommendation.

5.1.3 Comparison

To evaluate H-SAGE, we selected eight state-of-the-art methods from various relevant implementation and application

[14] Relation-Assurance-Value Factor

[15] A Dataset for Linking Knowledge Bases with Recommender Systems

[16] https://wiki.dbpedia.org/services-resources/ontology

[17] https://doi.org/10.17632/bszht7j9hd.1

[18] Click Through Rate Prediction

[19] Area Under the roc (Receiver Operating Characteristic) Curve & Accuracy

[20] Precision@K, Recall@K, Normalized Discounted Cumulative Gain @K

Table 2 The settings of hyper parameters wrt the experimental Environment. Literals: μ – Dropout Ration, b – Batch Size, η – Learning Rate, λ – L_2 Regularization Weight, ε – KGE Weight, s – hop-length wrt path-steps, d – Dimensions of Embedding, K – The sampling size of influential neighboring

Datasets	Environment Setting
Amazon-Book	$\mu=[-0.2, 08]$, $b=1024$, $\eta=7\times10^{-4}$, $\lambda=10^{-7}$, $\varepsilon=10^{-3}$, $s=3$, $d=32$, $K=16$
Last-FM	$\mu=[-0.2, 08]$, $b=1024$, $\eta=7\times10^{-4}$, $\lambda=10^{-7}$, $\varepsilon=10^{-3}$, $s=4$, $d=64$, $K=32$
Bing-News	$\mu=[-0.2, 08]$, $b=1024$, $\eta=7\times10^{-4}$, $\lambda=10^{-7}$, $\varepsilon=10^{-3}$, $s=4$, $d=64$, $K=32$

Table 3 CTRP Results: Evaluated wrt AUC and Acc. Terms: Upper-Bound (α), Lower-Bound (β), Mean ($\bar{x}$)

Approaches		Amazon-Book		Last-FM		Bing-News	
		AUC	Acc	AUC	Acc	AUC	Acc
PER		0.6210	0.5812	0.6129	0.5866	0.5153	0.4932
CKE		0.6419	0.6056	0.7389	0.6632	0.5432	0.5011
MCRec		0.6421	0.6287	0.7412	0.6701	0.5814	0.5633
RippleNet		0.6646	0.6419	0.7611	0.6787	0.6418	0.6002
KGAT		0.6789	0.6498	0.7691	0.6823	0.6796	0.6437
AKGE		0.6641	0.6399	0.7785	0.6891	0.6632	0.6411
NACF		0.7063	0.6734	0.7913	0.7232	0.6952	0.6741
DKEN		*0.7309**	*0.6911**	*0.8019**	*0.7415**	*0.7215**	*0.6959**
H-SAGE		**0.7597**	**0.7208**	**0.8313**	**0.7727**	**0.7436**	**0.7189**
Improved: (%)-age	α	03.79	04.12	03.54	04.04	02.97	03.20
	β	03.94	04.30	03.67	04.21	03.06	03.31
	$\bar{x}$	**03.87**	**04.21**	**03.60**	**04.12**	**03.02**	**03.25**

The numbers in bold represent the most significant values among the identical comparing outcomes, and the numbers in italic with '' describe the second important values accordingly

perspectives. **PER** [8] designates user to item and item to item relations and represents the heterogeneity of KG based on features extraction among nodes via Meta path interconnections. **CKE** [18] formulates aggregated Bayesian framework to empower matrix factorization with the help of TransR-based embeddings and combines this knowledge base with CF-mechanism for recommendation. **MCRec** [13] co-attentively supports HIG and attains the context and actual representations of entities via meta-path-based random walk technique. **RippleNet** [23] aggregates path and embedding-based recommendation methods and enriches the representations of users by adding the representations of items to the paths interconnecting users via propagation. **KGAT** [25] utilizes TransR to obtain the initial representations of users and items and performs representations propagation. It enriches entity representations with the information of corresponding neighbors of items. **AKGE** [26] models higher order relations though information propagation on less-distance-similarity-based self-constructed subgraph. **NACF** [27] collects neighborhood information of the entities to mine the potential relations between users 0an0d items to generate the potential preferences for their unobserved items. **DKEN** [28] exploits the impact of data exchange between implicit and explicit semantics of information in user to item interactions and KG-based features respectively, through CIS[21] layer to preserve a better grip on semantical and hierarchical structure of the information in KG.

5.1.4 Parameterization

We tried hyper-parameters wrt different tentative settings, and finalized an optimized environment of parameterization via Grid-Search technique [46] as summarized in Table 2; literals are defined in the caption of Table 2. We maintained μ between −0.2 to 0.8 and $b=1024$ for all datasets. Moreover, we selected optimized parameter-values for all datasets from the collections of candidate values, as $\eta=7\times10^{-4}$ out of $\{5\times10^{-2}, 5\times10^{-3}, 7\times10^{-3}, 7\times10^{-4}\}$, $\lambda=10^{-7}$ out of $\{10^{-10}, 10^{-9}, \ldots, 10^{2}, 10^{3}\}$, and $\varepsilon=10^{-3}$ out of $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Similarly, we kept s as 3, 4, 4, d as 32, 64, 64, and K as 16, 32, 32, for Amazon-Book, Last-FM and Bing-News respectively. We kept d unchanged for the implementation of baseline methods and utilized grid-search technique to adjust rest of their parameters. We repeated the experiments thrice and reported averages of the achieved results.

[21] Cross Information Share

Table 4 Top-k Recommendations via Prec, Rec & NDCG. Terms: Upper-Bound (α), Lower-Bound (β), Mean ($\bar{x}$)

Datasets & Evaluation @K=5, 10		Comparison and Improvements of H-SAGE wrt the Baseline Approaches											
		PER	CKE	MCRec	RNet[a]	KGAT	AKGE	NACF	DKEN	H-SAGE	Improved: (%)-age		
											α	β	$\bar{x}$
Amazon-Book	Prec@05	0.057	0.060	0.067	0.074	0.077	0.078	0.090	*0.095**	**0.110**	13.64	15.79	**14.71**
	Prec@10	0.052	0.050	0.060	0.076	0.080	0.080	0.088	*0.094**	**0.105**	10.48	11.70	**11.09**
	Rec@05	0.027	0.031	0.030	0.032	0.035	0.035	0.039	*0.040**	**0.042**	04.76	05.00	**04.88**
	Rec@10	0.029	0.032	0.034	0.035	0.037	0.040	0.043	*0.045**	**0.048**	06.25	06.67	**06.46**
	NDCG@05	0.031	0.031	0.031	0.032	0.033	*0.034**	*0.034**	*0.034**	**0.036**	05.56	05.88	**05.72**
	NDCG@10	0.032	0.032	0.034	0.035	*0.037**	0.035	*0.037**	*0.037**	**0.039**	05.13	05.41	**05.27**
Last-FM	Prec@05	0.057	0.060	0.067	0.064	0.073	0.075	0.073	0.077*	**0.081**	04.94	05.19	**05.07**
	Prec@10	0.050	0.053	0.060	0.061	0.064	0.064	0.066	0.068*	**0.071**	04.23	04.41	**04.32**
	Rec@05	0.020	0.030	0.040	0.050	0.050	0.040	0.050	0.053*	**0.058**	08.62	09.43	**09.03**
	Rec@10	0.030	0.040	0.050	0.060	0.070	0.060	0.080*	0.077	**0.091**	12.09	13.75	**12.92**
	NDCG@05	0.030	0.033	0.045	0.045	0.050	0.045	0.055	0.057*	**0.063**	09.52	10.53	**10.03**
	NDCG@10	0.060	0.080	0.090	0.100	0.110	0.114	0.119*	0.113	**0.130**	08.46	09.24	**08.85**
Bing-News	Prec@05	0.004	0.004	0.006	0.007	0.007	0.007	0.009	*0.010**	**0.011**	09.09	10.00	**09.55**
	Prec@10	0.004	0.004	0.006	0.007	0.008	0.008	*0.010**	*0.010**	**0.011**	09.09	10.00	**09.55**
	Rec@05	0.027	0.027	0.028	0.032	0.035	0.035	0.039	*0.040**	**0.045**	11.11	12.50	**11.81**
	Rec@10	0.028	0.029	0.030	0.035	0.037	0.036	0.043	*0.045**	**0.055**	18.18	22.22	**20.20**
	NDCG@05	0.010	0.011	0.015	0.017	0.023	0.025	0.029	*0.033**	**0.037**	10.81	12.12	**11.47**
	NDCG@10	0.050	0.070	0.100	0.110	0.110	0.115	0.120	*0.125**	**0.132**	05.30	05.60	**05.45**

[a]RippleNet. *The numbers in bold represent the most significant values among the identical comparing outcomes, and the numbers in italic with '*' describe the second important values accordingly

5.2 Comparative study (RQ1)

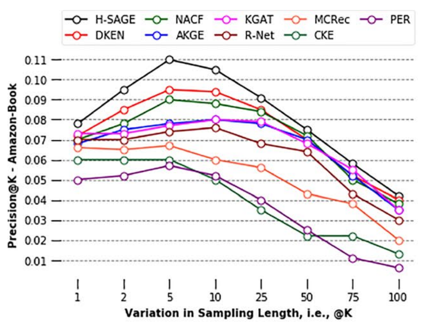
Formally, Table 3 represents the complete results of CTRP wrt AUC and Acc, whereas Table 4 demonstrates the results of top-K recommendations wrt (Prec, Rec and NDCG)@K=5 and 10 only. Regarding top-K recommendation, we clarify that we performed experiments on eight different variations of K, i.e., 1, 2, 5, 10, 25, 50, 75, 100, but due to the space limitations, we could only present the results via K=5 and 10 in Table 4, and the complete outcome of this process is shown in Fig. 3.

In this section, we discuss the results analysis of the conducted experiments wrt the comparative study. The proposed approach has outperformed the baselines on all datasets with decent margin of improvement, shows that H-SAGE is capable of providing effective recommendations due to its strong mechanisms of data filtration and information-hashing.

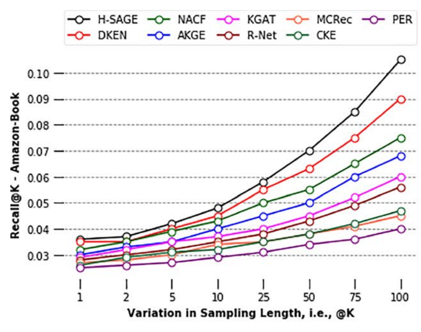
H-SAGE preserves meaningful information via dedicated meta-paths-based on the inter-entities semantic relevance. It relies on semantic relevance instead of depending on short distance or fixed weights among entities. In DKEN [28]; first, there is no mechanism to filter out the irrelevant information. Second, redundant data is provided to two parallel layers simultaneously. Third, handling more instances of data on different places increases the computational overhead. Although it can cause space complexity, currently it's not a big issue on smaller datasets. In NACF [27], attention mechanism is used to assign weights in the subgraph; and similarities among them are calculated based on the assigned weights, regardless of the syntactic or semantic relevance among entities or their mutual relations. NACF is outperformed by DKEN notifying that incorporation of CIS and KEN layers contributed more to the performance of DKEN during distribution and aggregation of information. Likewise, Generalization-Layer[22] enhanced the generalization capability in learning high level information features from the data. During the construction of subgraph in AKGE [26], the stance *lesser is the distance between two entities in the Euclidian space, greater is the similarity between them* is considered as the basic rule to find the similarity among entities in KG, caused the emergence of noise to the information.

In KGAT [25], extensive propagation is performed and information is gathered without the application of any noise filtration constraint caused noise emergence to the local knowledge base. Moreover, KGAT fails to efficiently attain the hierarchical and sequential structure of nodes in KG due to its attentive-embedding-propagation

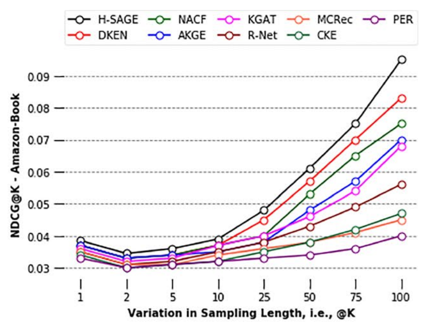
[22] DNN Layer is Generalization Layer in DKEN



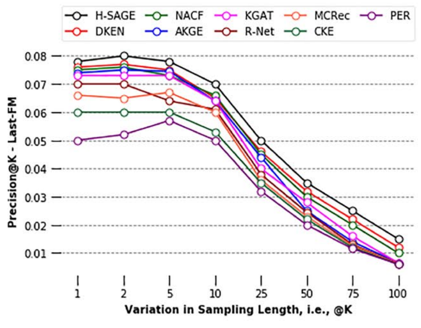
(a) Prec@K on Amazon-Book



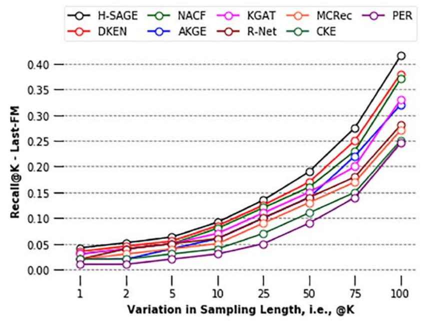
(b) Rec@K on Amazon-Book



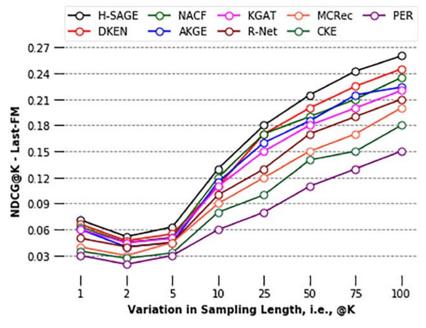
(c) NDCG@K on Amazon-Book



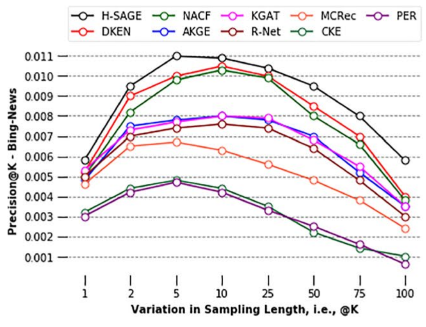
(d) Prec@K on Last-FM



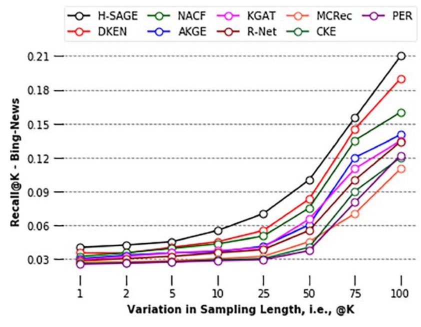
(e) Rec@K on Last-FM



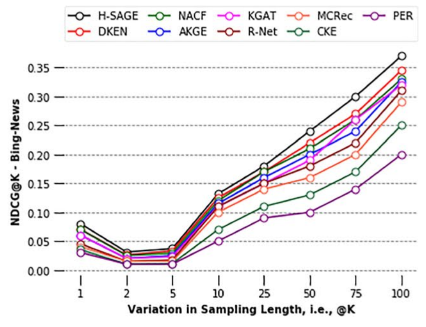
(f) NDCG@K on Last-FM



(g) Prec@K on Bing-News



(h) Recall@K on Bing-News



(i) NDCG@K on Bing-News

Fig. 3 Result-Analysis of top-*K* recommendations wrt (Prec, Rec, NDCG)@K on the three Mentioned Datasets

– a reason that AKGE outperformed it. But, both AKGE and KGAT are outperformed by NACF in about all cases, justifying that the incorporation of similarity mechanism is in favor of performance. Moreover, the propagation of attentive embedding and aggregation is capable of generalizing the corresponding frameworks smoothly as compared to the straight-propagation and aggregation, and the meta-path-based random walk traversal techniques in RippleNet [23] and MCRec [13] respectively. It is why AKGE and KGAT outperformed RippleNet and MCRec wrt different aspects. Counterwise, MCRec, CKE [18] and PER [8] are outperformed by RippleNet in majority of their comparisons intensifying the supremacy of information propagation over the limited meta-path-based or simple embedding and regularization-based methods. Further, compared to CKE and PER, MCRec comparatively performed well highlighting the significance of information selection technique of Meta-path-based methods over simple regularization or embedding-based methods. Finally, although it's evident from the empirical values that CKE has a tiny upper edge over PER in some cases, their average performance is almost indistinguishable according to the experiments.

Table 5 Study of Time Complexity Comparison of the proposed Approach with the baseline approaches. Terms used: M – Modules, L – Layers, T – Type of the mentioned item, X – No. of Modules or Layers

Approaches	M / L Mentioned			Defined Time Complexity	Time Complexity of M / L			
	No	Yes			$\geq O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$
		T	X					
PER	✓			$O(mn^2)$	✓			
CKE	✓			Not Defined	✓			
MCRec	✓			$O(lLN + lN \log N)$	✓			
RippleNet		M	2	$O(YHK^H d^2 + Gd^2)$ $O(YHK^{H+1} d + YHK^H d^2)$	2			
KGAT		M	3	$O\left(\lvert G_2\rvert d^2 + \sum_{i=1}^{L} \lvert G\rvert d_i d_{i-1} + \lvert G\rvert d_i\right)$	1	1	1	
AKGE		M	3	$O(PQ + QlogQ)$ $O\left(\lvert R\rvert \sum_{k=1}^{K} \lvert E_s\rvert d^2\right)$	1	1	1	
NACF	✓			Not Defined	✓			
DKEN		L	4	Not Defined	1	2	1	
H-SAGE$_{\mathbf{Miss}}$		L	2	$\lvert M\rvert \sum_{s=1}^{S} \lvert \bar{G}_s\rvert d_s^2 \simeq O\left(n^2\right)$ $O\left(\sum_{i=1}^{3} \lvert \mathrm{H}\rvert b_i\right) \simeq O(n)$		1	1	
H-SAGE$_{\mathbf{Hit}}$		L	2	$\lvert M\rvert \sum_{s=1}^{S} \lvert \bar{G}_s\rvert d_s^2 \simeq O\left(n^2\right)$ $O(1)$		1		1

5.3 Complexity study (RQ2)

We tabularized the comparison of all approaches wrt the executional time-complexity, as shown in Table 5. For instance, PER and MCRec defined complexities but not defined modules or layers, so, we only put ticks under *M / L Mentioned* and mentioned their complexities under *Defined Time Complexity*. We put information of CKE, NACF and DKEN based on their defined pseudocodes; CKE and NACF neither defined complexities nor mentioned modules or layers; whereas, DKEN not defined its complexity but mentioned its layers. Moreover, RippleNet, KGAT and AKGE defined complexities as well as mentioned their modules. Analysis demonstrates that the overall complexities of PER, CKE, MCRec and NACF are greater than $O(n^2)$.

Similarly, in both modules RippleNet and in one M/L KGAT, AKGE and DKEN each, undergo TC greater than $O(n^2)$. Moreover, one M/L of KGAT, AKGE and H-SAGE each, and two of DKEN possess TC of $O(n^2)$; and one M/L of KGAT, AKGE, DKEN and H-SAGE undergo TC of $O(n)$. Formally, the worst TC of H-SAGE, i.e., $O(n^2)$, is experienced by its first layer due the calculation of similarity; but, its second layer can experience time complexity of $O(1)$ in case of Hash-Hit, otherwise $O(n)$ in Hash-Miss. Conclusively, H-SAGE has the minimum contribution to Table 5, we can claim that it has outperformed the-state-of-the-art methods in comparison of the computational time complexity.

5.4 Sparsity study (RQ3)

The experimental results (i.e., Tables 3 and 4, and Fig. 4) demonstrate that the performance of H-SAGE is not bad on sparse datasets as well. For AUC in Fig. 4, the performance continuously increased up to 70% of data-utilization on all datasets, where H-SAGE attained the highest performance. After this till the end, the performance on all datasets decreased gradually except Amazon-Book that abruptly fell down after 90% utilization of data, as shown in Fig. 4(a). On the other hand, it is evident from Fig. 4(b) that the error-rate wrt AUC is considered as the additive inverse of the performance of AUC. Similarly, for Acc, the performance on Amazon-Book and Bing-News gradually increased up to 70% of the data where the performance is highest. However, on Last-FM from 20 to 55% of the data, the Acc kept variating, but achieved the highest performance on 60%. After the highest values, the performance wrt other datasets decreased gradually except Bing-News that abruptly degraded after 80% of data, as in Fig. 4(c). It implies that while dealing with larger sizes of information of news titles and snippets, the model loosed its grip on information structure that caused overfitting. Contrariwise, from Fig. 4(d), the error-rate on Acc is additive inverse of its performance.

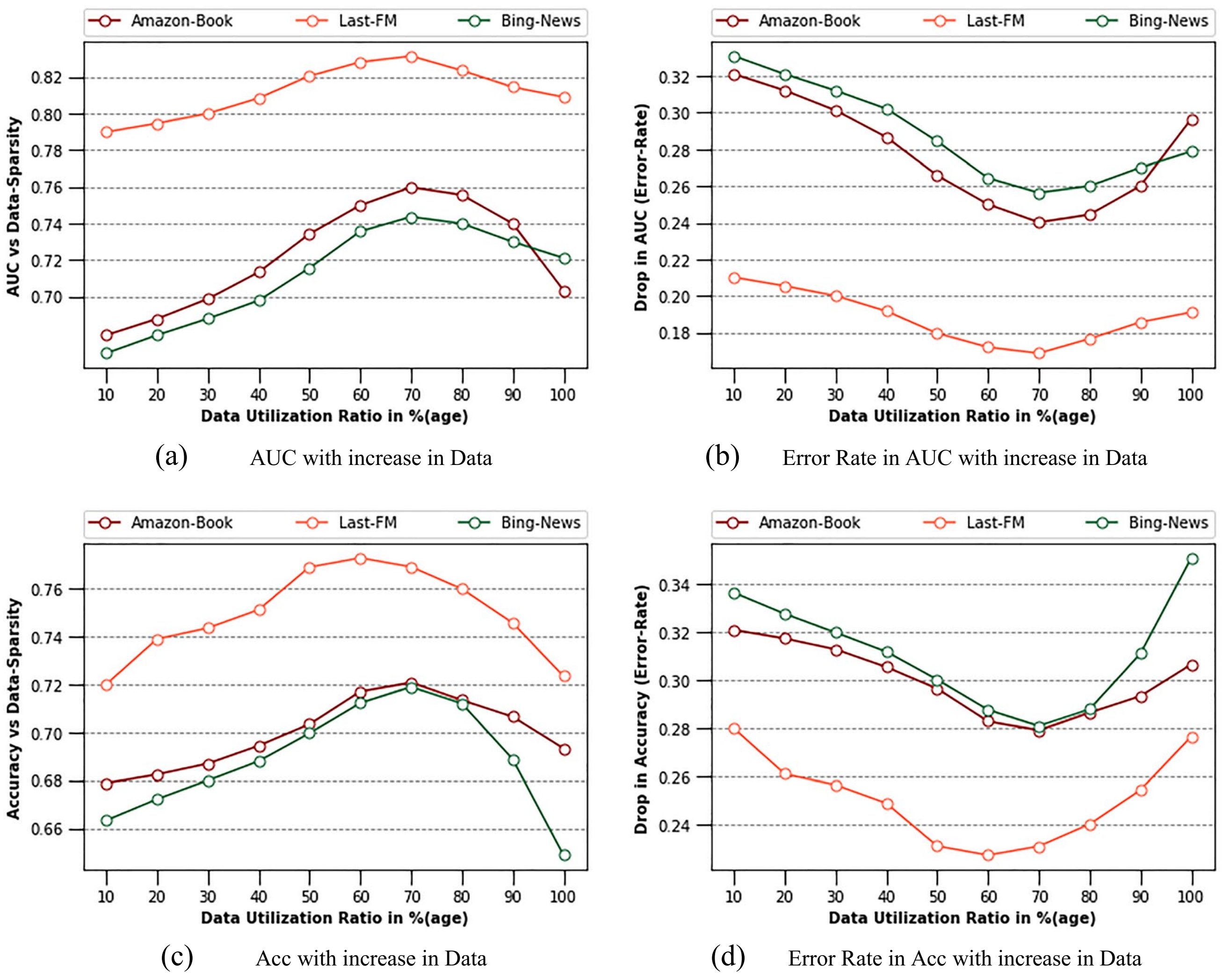


Fig. 4 AUC and Acc results analysis wrt the Data-Sparsity on the proposed datasets

Table 6 The comparison of performance among variations of H-SAGE based on path selection techniques

H-SAGE Variants	Amazon-Book		Last-FM		Bing-News	
	AUC	Acc	AUC	Acc	AUC	Acc
H-SAGE$_{MS}$	0.6832	0.6601	0.7532	0.6786	0.6656	0.6515
H-SAGE$_{MR}$	0.7011	0.6689	0.7609	0.6910	0.6708	0.6602
H-SAGE$_{MW}$	0.7219	0.6799	0.7723	0.6987	0.6898	0.6692
H-SAGE$_{SN}$	0.7342	0.6957	0.8011	0.7235	0.7029	0.6720
H-SAGE	**0.7597**	**0.7208**	**0.8313**	**0.7727**	**0.7436**	**0.7189**

The numbers in bold represent the most significant values among the comparing outcomes

Finally, we claim that H-SAGE is capable to effectively deal with the limitations of data sparsity.

5.5 Ablation study (RQ4)

In section, we demonstrate the ablation study of H-SAGE wrt two different applicability variations. First, wrt the importance of information relevance; and second, wrt the significance of H-SAGE's architecture.

5.5.1 Impact of prominent Meta-paths

We present the comparison of the proposed *Node Relevance Guided-walk* (NRG) with four state-of-the-art path modeling

Table 7 The comparison of performance wrt different variations of H-SAGE. Abbreviations: H-SAGE$_{LS}$ is H-SAGE through LS (Locality Sensitive-hashing), LH (Learning to Hash), NR (Node Relevance), H-SAGE$_{LS+LH}$ H-SAGE LS and LH and so on, & H-SAGE means H-SAGE$_{ALL}$ i.e., NR + LS + LH

H-SAGE Variants	Amazon-Book		Last-FM		Bing-News	
	AUC	Acc	AUC	Acc	AUC	Acc
H-SAGE$_{LS}$	0.6875	0.6578	0.7698	0.7192	0.6912	0.6434
H-SAGE$_{LH}$	0.7099	0.6626	0.7768	0.7278	0.7007	0.6508
H-SAGE$_{LS+LH}$	0.7120	0.6791	0.7811	0.7332	0.7065	0.6621
H-SAGE$_{NR+LS}$	0.7331	0.6902	0.8023	0.7519	0.7203	0.6842
H-SAGE$_{NR+LH}$	0.7482	0.7099	0.8134	0.7589	0.7289	0.6923
H-SAGE	**0.7597**	**0.7208**	**0.8313**	**0.7727**	**0.7436**	**0.7189**

The numbers in bold represent the most significant values among the comparing outcomes

Table 8 The result analysis of H-SAGE's performance wrt the Meta-path length of higher-order relations

s	Amazon-Book		Last-FM		Bing-News	
	AUC	Acc	AUC	Acc	AUC	Acc
1	0.7326	0.7023	0.8091	0.7485	0.7101	0.6819
2	0.7506	0.7179	0.8259	0.7622	0.7298	0.7011
3	**0.7597**	**0.7208**	0.8309	**0.7727**	**0.7436**	0.7181
4	0.7547	0.7122	**0.8313**	0.7681	0.7388	**0.7189**
5	0.7101	0.6599	0.7546	0.6697	0.6755	0.6257

The numbers in bold represent the most significant values among the comparing outcomes

methods. First, we applied *Meta-path guided Similarity* (MS) to create subgraph with similar Meta-paths-based on their mutual similarity [11]. Second, we used *Meta-path guided Retrieval* (MR) technique to construct local subgraph based on retrieval of selective meta-paths [12]. Third, we utilized *Mata-path guided random Walk* (MW) approach to acquire prominent paths from KG to create the subgraph [15]. Fourth, we used *Shortest-distance steered Node-selection* (SN) method to access salient paths to construct subgraph [26].

We experimented the constructed subgraphs via H-SAGE under the titles of H-SAGE$_{MS}$, H-SAGE$_{MR}$, H-SAGE$_{MW}$, H-SAGE$_{SN}$ and H-SAGE$_{NRG}$ (H-SAGE), and summarized the results in Table 6. The achieved outcomes express small variations with minimal increase in the results from SAGE$_{MS}$ to SAGE$_{MW}$ via SAGE$_{MR}$. However, the SAGE$_{SN}$ is a bit better than the previous techniques due to the utilization of shortest-distance-based similarity. The experimental results demonstrate that H-SAGE has outperformed the state-of-the-art methods by clarifying that the relevance-based information collection can better contribute to the performance.

5.5.2 Impact of different modules

For an easy conduct, we verbalize H-SAGE wrt different modules as H-SAGE$_{LS}$, H-SAGE$_{LH}$, H-SAGE$_{LS+LH}$, H-SAGE$_{NR+LS}$, H-SAGE$_{NR+LH}$, H-SAGE$_{NR+LS+LH}$ i.e., H-SAGE. In case of H-SAGE$_{LS}$, the performance is lowest because the locality-based hashing alone can neither access the higher order relations nor effectively maintain the required graph structure. H-SAGE$_{LH}$ – the standalone learning to hash – is however better than LS wrt the achieved performance, however it cannot effectively tackle the graph structure in hash-miss. Moreover, H-SAGE$_{LS+LH}$ has effectively tackled the hash-miss and produced better performance as compared to LS or LH but still it is not satisfactory. The main reason of performance degradation is the occurrence of noise and irrelevant data in the underlying information.

Next, we applied H-SAGE$_{NR+LS}$ and H-SAGE$_{NR+LH}$, one by one, to judge the difference in their results. Amazingly via H-SAGE$_{NR+LS}$, the performance is better than that of H-SAGE$_{LS+LH}$; and more amazingly, the results of H-SAGE$_{NR+LH}$ are better than those of H-SAGE$_{NR+LS}$. At this point, we tried a collection of all, i.e., H-SAGE$_{NR+LS+LH}$, and achieved extraordinarily better performance compared to the previously discussed modules and their combinations with H-SAGE, as summarized in Table 7. Thus, we confirmed H-SAGE$_{NR+LS+LH}$ as the proposed model, i.e., H-SAGE with the set of NR + LS + LH as the necessary modules.

5.6 Sensitivity study (RQ5)

In this section, we demonstrate the impact of hyper-parameter's sensitivity on the performance of H-SAGE.

5.6.1 Length of Meta-path in higher-order relations

We evaluated the performance of H-SAGE based on the increase in hop-length s wrt the Meta-paths. We performed experiments on $s = 1$ to 5 with increase of 1 in s at each next

Table 9 The result analysis of H-SAGE's performance wrt the embedding length of entity representations

ℓ	Amazon-Book		Last-FM		Bing-News	
	AUC	Acc	AUC	Acc	AUC	Acc
2^1	0.7124	0.6675	0.7839	0.7314	0.6892	0.6723
2^2	0.7299	0.6835	0.8036	0.7497	0.7036	0.6860
2^3	0.7432	0.7018	0.8199	0.7536	0.7212	0.7011
$\mathbf{2^4}$	0.7521	0.7156	**0.8313**	0.7622	0.7376	0.7127
$\mathbf{2^5}$	**0.7597**	**0.7208**	0.8311	0.7724	**0.7436**	**0.7189**
$\mathbf{2^6}$	0.7501	0.7125	0.8232	**0.7727**	0.7388	0.7117
2^7	0.6855	0.6432	0.7623	0.7109	0.6645	0.6433

The numbers in bold represent the most significant values among the comparing outcomes

Table 10 The result analysis of H-SAGE's performance wrt the sampling length of influential neighboring

K	Amazon-Book		Last-FM		Bing-News	
	AUC	Acc	AUC	Acc	AUC	Acc
2^1	0.7367	0.6731	0.7823	0.7278	0.7065	0.6802
2^2	0.7432	0.6891	0.8067	0.7412	0.7208	0.6898
2^3	0.7509	0.7030	0.8235	0.7546	0.7316	0.7032
$\mathbf{2^4}$	**0.7597**	0.7156	**0.8313**	0.7622	0.7376	0.7127
$\mathbf{2^5}$	0.7552	**0.7208**	0.8311	0.7724	**0.7436**	**0.7189**
$\mathbf{2^6}$	0.7501	0.7125	0.8232	**0.7727**	0.7388	0.7117
2^7	0.6835	0.6445	0.7456	0.6875	0.6643	0.6457

The numbers in bold represent the most significant values among the comparing outcomes

iteration. In the results, we noticed a continuous and rapid increase in the performance up to $s=3$. With a further increase in s, i.e., $s=4$, the performance of H-SAGE still remained better but undergone a slight downfall compared to that on $s=3$ wrt a few instances. However, on $s>4$; the performance faced such an extraordinary downfall that on $s=5$, H-SAGE undergone the worst performance, as shown in Table 8. Hence, we can conclude that on $s>4$, the framework faces overfitting that hinders it in effectively capturing the graph structure.

5.6.2 Length of embedding dimensions

The levels of embeddings are shown through $\ell \mid \ell = 2^d$ where $d=1, 2, \ldots, 7$. We performed the experiments wrt $d=1$ to 7 with an increase of 1 in d for each next iteration. From the results, we noticed that with the increase in d, the performance is continuously and rapidly increasing up to $d=5$, where it acquired the highest performance as compared to the most of the incurred instances. With the next increase in d, i.e., on $d=6$, though the performance of H-SAGE is still in a better and persisting position, a trivial decrease is undergone wrt a sound number of instances in the observations. However, on $d>6$, a significant degradation is observed, and on $d=7$, H-SAGE acquired the worst performance. It means that H-SAGE can generalize the representations effectively up to $d=6$ only. Therefore, we can conclude that after $d=6$, the framework has lost its grip on information structure, considered the triplet's false granularity as valid, included that irrelevant data to the information and degraded the performance. Finally, we summarized the performance analysis of H-SAGE in Table 9.

5.6.3 Sampling length of influential neighboring

We represent the levels of neighborhood sampling through $K \mid K=2^i$ where $i=1, 2, \ldots, 7$. We performed the experiments wrt $i=1$ to 7 with an increase of 1 in i for each next iteration. From the results, we observed that with increase in i, the performance of H-SAGE is continuously and rapidly increasing up to $i=5$, and preserved an optimal performance on $i=4$, 5 and 6 wrt the sampling size. However, on $i>6$; a significant degradation is observed in the performance, and on $i=7$; the model acquired the worst performance. The conclusion is similar to that of Section 5.6.2, and we summarized the results in Table 10.

5.7 Case study (RQ6)

H-SAGE is capable of providing satisfactory explanations about its generated recommendations. To further explain its working mechanism, we present a daily life recommendation scenario from *Bing-News*; that provides news suggestions to the users based on their previous-interactions with the online news catalogues via click-record. We retrieved the following

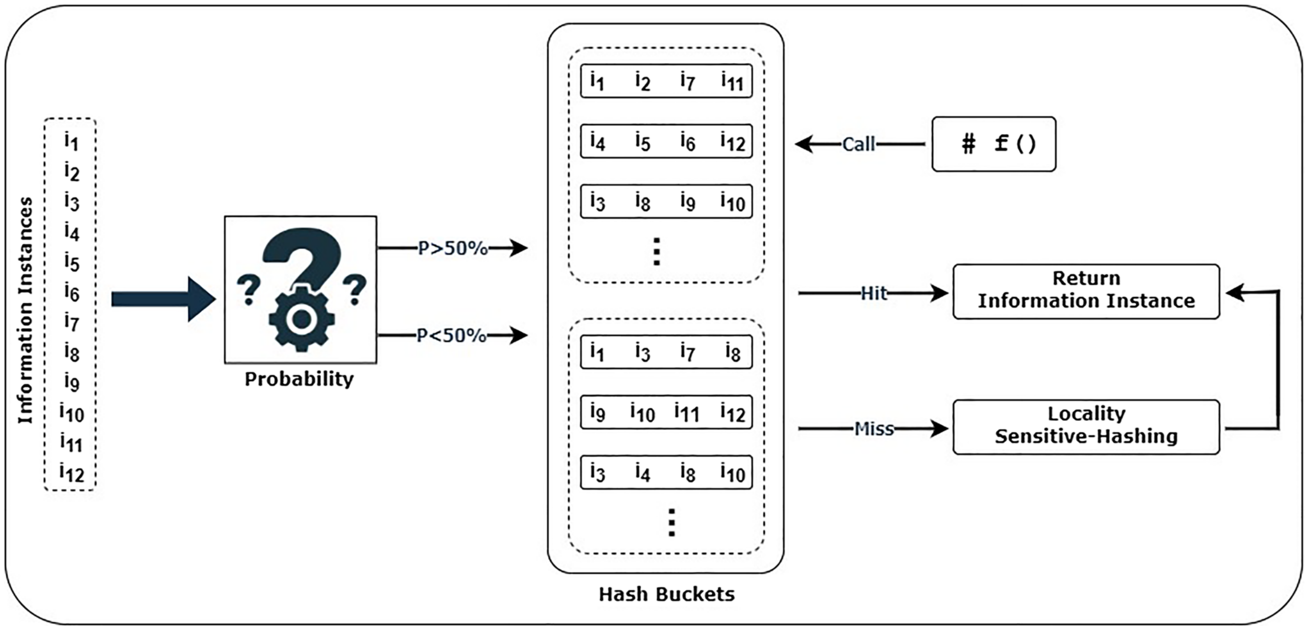


Fig. 5 Case Study of a daily-based news recommendation scenario

seven random click-samples from the user's interaction-logs and presented below:

C1: None is safe: Osaka, Japan also crumples under **COVID-19 onslaught.**
C2: Biden's **COVID warning**: "Unvaccinated will end up paying the **price**"
C3: Biden's remarks announcing Afghanistan **troop withdrawal**.
C4: US announces plans to **cut troop levels** in Afghanistan.
C5: Trump backs **Afghanistan withdrawal**, putting him at odds with some Republicans.
C6: **COVID-19** pandemic **effected** the US **economy** badly.
C7: **Economy grew** at 6.4% in 1st quarter of 2021 – the massive **vaccination rollout** and the **army withdrawal**.

According to [50], we considered the heading-entities as instances in the text of the retrieved click samples, and indexed as i_1 = "COVID-19 onslaught", i_2 = "COVID warning", i_3 = "price", i_4 = "troop withdrawal", i_5 = "cut troop level", i_6 = "Afghanistan withdrawal", i_7 = "COVID-19", i_8 = "affected", i_9 = "economy", i_{10} = "Economy grew", i_{11} = "vaccination rollout" and i_1 = "army withdrawal". After the required preprocessing, we categorized these instances into the semantically relevant hash-buckets-based on their likelihood, as discussed in Section 4.2.2. We tackled the information in the stated buckets through function calls. In case of hash-hit, the target hash-code is returned, otherwise locality sensitive hashing technique is applied to acquire the required hash-codes around the location of hash-miss in the buckets. The working phenomenon of the case study is portrayed in Fig. 5. Formally, in response to the above click history, stating from the random system's outcome, top three of the candidate news are copied and pasted below. Although the framework of H-SAGE has smoothly generated the news recommendations, it is quite evident from the system's response that the performance still needs a sound improvement. For instance, it is well clear that N1 is normal, N2 is ambiguous, whereas N3 is totally based on unrealistic assumption, as shown below.

N1: US announces troop withdrawal from Afghanistan; considering an economy overload.
N2: COVID: Bad to economy or Good; currently undecidable.
N3: US announces troop withdrawal, due to COVID-19.

Therefore, based on demonstration of the case study wrt the working mechanism, we can conclusively narrate that H-SAGE is capable of providing satisfactory explanations about its generated recommendations.

6 Conclusion and future work

In this work, we proposed a novel semantic-relevance-and-hashing guided KGE enhancement approach for recommendation. We introduced *Node Relevance Guided-walk* (NRG) modeling technique to construct entity-relevance-based influential graph by capturing higher-order semantically relevant nodes in KG. We converted the graph to hash-codes for implementation. We proposed *dProb* to place hash-codes in identical hash-buckets-based on their mutual likelihood to maximize the Hash-Hits. We also used *dProb* to generate feasible hash function-calls. For Hash-Miss, we applied LS-hashing to extract the required hash-codes from the target hash-bucket around the location of Hash-Miss and return the information. Finally, we used a predictive interface to evaluate the retrieved hash-codes, compute the preferences and generate the formal recommendation responses. We evaluated H-SAGE on three real world datasets and compared its performance and time-complexity with eight baseline methods. The experimental and theoretical analysis signifies that H-SAGE has outperformed the-state-of-the-art methods.

In future work, we plan to exploit KGEE for drugs and precautionary alerts recommendation against any happened or happening disease/pandemic's outspread. Further, we plan to enhance KGE-based retrieval of semantically interlinked entities and relations to further preserve actual information structure of the triplets.

Acknowledgments This research work was supported in part by the Basic Research Program of Jiangsu Province (BK20191274) and the National Natural Science Foundation of China (62176121, 61772269).

Authors contribution All authors have actively participated in the literature analysis, the interpretation of results and preparation of final manuscript; where, **Nasrullah Khan:** Conceptualization, Methodology, Data Curation, Implementation, Writing – Original draft preparation. **Zongmin Ma:** Supervision, Validation, Writing – Review and Editing. **Li Yan:** Validation, Writing – Review and Editing. **Aman Ullah:** Validation, Investigation, Software. All authors have read and approved the final manuscript.

Declaration

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

References

1. Bobadilla J, González-Prieto A, Ortega F, Lara-Cabrera R (2021) Deep learning feature selection to unhide demographic recommender systems factors. Neural Comput & Applic 33:7291–7308
2. Sheng QZ et al (2021) Hetegraph: graph learning in recommender systems via graph convolutional networks. Neural Comp Appl 1–17
3. Singh PK, Sinha M, Das S, Choudhury P (2020) Enhancing recommendation accuracy of item-based collaborative filtering using bhattacharyya coefficient and most similar item. Appl Intell 50:4708–4731
4. Sang L, Xu M, Qian S, Wu X (2011) Knowledge graph enhanced neural collaborative recommendation. Expert Syst Appl 164:113992
5. Ahmadian S, Meghdadi M, Afsharchi M (2018) Incorporating reliable virtual ratings into social recommendation systems. Appl Intell 48:4448–4469
6. Sun Z et al (2019) Research commentary on recommendations with side information: a survey and research directions. Electron Commer Res Appl 37:100879
7. Nisha C, Mohan A (2019) A social recommender system using deep architecture and network embedding. Appl Intell 49:1937–1953
8. Yu X et al. (2014) Personalized entity recommendation: A heterogeneous information network approach. In Proceedings of the 7th ACM international conference on Web search and data mining, 283–292
9. Wang S, Du Z, Ding M, Rodriguez-Paton A, Song T (2021) Kg-dti: a knowledge graph based deep learning method for drug-target interaction predictions and alzheimer's disease drug repositions. Appl Intell 1–12
10. Ma M, Na S, Wang H, Chen C, Xu J (2021) The graph-based behavior-aware recommendation for interactive news. Appl Intell 1–17
11. Shi C et al. (2015) Semantic path based personalized recommendation on weighted heterogeneous information networks. *In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 453–462
12. Zhao H, Yao Q, Li J, Song Y, Lee DL (2017) Meta-graph based recommendation fusion over heterogeneous information networks. In Proceedings of the 23rd ACM SIGKDD int. conference on knowledge discovery and data mining, 635–644
13. Hu B, Shi C, Zhao WX, Yu P (2018) Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 1531–1540
14. Zhao W, Wu R, Liu H (2016) Paper recommendation based on the knowledge gap between a researcher's background knowledge and research target. Inf Process Manag 52(5):976–988
15. Shi C, Hu B, Zhao WX, Philip SY (2018) Heterogeneous information network embedding for recommendation. IEEE Trans Knowl Data Eng 31:357–370
16. Sun Z et al. (2018) Recurrent knowledge graph embedding for effective recommendation. In Proceedings of the 12th ACM Conference on Recommender Systems, 297–305
17. Wang X et al (2019) Explainable reasoning over knowledge graphs for recommendation. Proc AAAI Conf Artif Intell 33:5329–5336
18. Zhang F, Yuan NJ, Lian D, Xie X, Ma W (2016) Collaborative knowledge base embedding for recommender systems. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 353–362
19. Huang J, Zhao WX, Dou H, Wen J, Chang EY (2018) Improving sequential recommendation with knowledge enhanced memory networks. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 505–514
20. Wang H, Zhang F, Xie X, Guo M (2018) Dkn: Deep knowledge-aware network for news recommendation. In Proceedings of the 2018 world wide web conference, 1835–1844
21. Xin X, He X, Zhang Y, Zhang Y, Jose J (2019) Relational collaborative filtering: Modeling multiple item relations for recommendation. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 125–134
22. Cao Y, Wang X, He X, Hu Z, Chua T (2019) Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In The world wide web conference, 151–161
23. Wang H et al (2019) Exploring high-order user preference on the knowledge graph for recommender systems. ACM Trans Inform Syst (TOIS) 37:1–26

24. Wang H et al. (2019) Knowledge graph convolutional networks for recommender systems. In The World Wide Web conference, pp. 3307–3313
25. Wang X, He X, Cao Y, Liu M, Chua T (2019) Kgat: Knowledge graph attention network for recommendation. In Proceedings of the 25th ACM SIGKDD Int. Conference on Knowledge Discovery & Data Mining, 950–958
26. Sha X, Sun Z, Zhang J (2019) Attentive knowledge graph embedding for personalized recommendation. arXiv preprint arXiv:1910.08288
27. Zhang D et al (2020) Neighborhood aggregation collaborative filtering based on knowledge graph. Appl Sci 10:3818
28. Guo X et al (2020) Dken: deep knowledge-enhanced network for recommender systems. Inf Sci 540:263–277
29. Wang M, Wu T, Qi G (2020) A hash learning framework for search-oriented knowledge graph embedding. In ECAI 2020, 921–928 (IOS Press)
30. Salakhutdinov R, Hinton G (2009) Semantic hashing. Int J Approx Reason 50:969–978
31. Zhang Y, Lin H, Yang Z, Li Y (2011) Neighborhood hash graph kernel for protein–protein interaction extraction. J Biomed Inform 44:1086–1092
32. Cao Z, Long M, Wang J, Yu P (2017) Hashnet: Deep learning to hash by continuation. In Proceedings of the IEEE international conference on computer vision, 5608–5617
33. Huang W, Li Q, Meng S (2020) Kg2rec: Lsh-cf recommendation method based on knowledge graph for cloud services. Wireless Networks 1–12
34. Datar M, Immorlica N, Indyk P, Mirrokni V (2004) Locality-sensitive hashing scheme based on p-stable distributions. In Proceedings of the twentieth annual symposium on Computational geometry, 253–262
35. Hansen C, Hansen C, Simonsen JG, Alstrup S, Lioma C (2020) Content-aware neural hashing for cold-start recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 971–980
36. Liu X et al. (2020) Reinforced short-length hashing. IEEE Trans Circ Syst Video Technol
37. Tan Q et al. (2020) Learning to hash with graph neural networks for recommender systems. In Proceedings of The Web Conference 2020, 1988–1998
38. Hui B, Zhang L, Zhou X, Wen X, Nian Y (2021) Personalized recommendation system based on knowledge embedding and historical behavior. Appl Intell 1–13
39. Ji G, He S, Xu L, Liu K, Zhao J (2015) Knowledge graph embedding via dynamic mapping matrix. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 687–696
40. Chen G, Wang F, Zhang C (2010) Collaborative filtering using orthogonal nonnegative matrix tri-factorization. Inf Process Manag 45:368–379
41. Zhou Y, Li D, Huo S, Kung S (2020) Soft-root-sign activation function. arXiv preprint arXiv:2003.00547
42. Ramachandran P, Zoph B, Le Q V (2017) Searching for activation functions. arXiv preprint arXiv:1710.05941
43. Dolker E et al (2018) Elastic net regularization in lorentz force evaluation. NDT E Int 99:141–154
44. Wang H, Xu Y (2021) Sparse elastic net multi-label rank support vector machine with pinball loss and its applications. Appl Soft Comput 104:107232
45. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2012) Bpr: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618
46. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
47. Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multirelational data. Adv Neural Inform Proc Syst 26
48. He R, McAuley J (2016) Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *In proceedings of the 25th international conference on world wide web*, 507–517
49. Cantador I, Brusilovsky P, Kuflik T (2011) Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011). *In Proceedings of the fifth ACM conference on Recommender systems*, 387–388
50. Qin T, Liu T (2013) Introducing letor 4.0 datasets. arXiv preprint arXiv:1306.2597

Nasrullah Khan acquired BS and MS in computer science from Gomal University Dera Ismail Khan, Pakistan, and Bahauddin Zakariya University Multan, Pakistan, respectively. He is a faculty member in Computer Science at COMSATS University Islamabad, Pakistan. Currently, he is pursuing PhD in Computer science with Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interest includes Knowledge Graph, Recommender Systems, Natural Language Processing and Mathematical Computation.

Dr. Zongmin Ma is currently a full professor at Nanjing University of Aeronautics and Astronautics, China. His research interests include big data and knowledge engineering, the Semantic Web, temporal/spatial information modeling and processing, deep learning, and knowledge representation and reasoning with a special focus on information uncertainty. He has published more than two hundred papers in international journals and conferences on these topics. In addition, he (co-)authors five monographs with Springer. He is a Fellow of the IFSA and a senior member of the IEEE.

Dr. Li Yan received her PhD degree from Northeastern University, China. She is currently a full professor at Nanjing University of Aeronautics and Astronautics, China. Her research interests mainly include big data processing, knowledge graph, spatiotemporal data management, and fuzzy data modeling. She has published more than fifty papers on these topics. She is the author of three monographs published by Springer.

Aman Ullah has completed his BS degree in computer science from Gomal University DIkhan, Pakistan, and MS degree in software engineering from Abasyn University Peshawar, Pakistan, in 2013 and 2017 respectively. He is currently pursuing PhD degree with School of Computer Science and Engineering, Central South University, Changsha, China. He has published several JCR indexed by SCI research papers in reputed journals. His current research interests include Complex Networks (structure and function of complex networks, -especially, Influential nodes identification, community detection, link prediction), Transparent & Parallel Computing, and Software Engineering (Software cost estimation and software testing).