Recommender systems based on graph embedding techniques: A comprehensive review

Yue Deng[a,*]

[a]*Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China, Chengdu 611731, China*

Abstract

As a pivotal tool to alleviate the information overload problem, recommender systems aim to predict user's preferred items from millions of candidates by analyzing observed user-item relations. As for alleviating the sparsity and cold start problems encountered by recommender systems, researchers generally resort to employing side information or knowledge in recommendation as a strategy for uncovering hidden (indirect) user-item relations, aiming to enrich observed information (or data) for recommendation. However, in the face of the high complexity and large scale of side information and knowledge, this strategy largely relies for efficient implementation on the scalability of recommendation models. Not until after the prevalence of machine learning did graph embedding techniques be a recent concentration, which can efficiently utilize complex and large-scale data. In light of that, equipping recommender systems with graph embedding techniques has been widely studied these years, appearing to outperform conventional recommendation implemented directly based on graph topological analysis (or resolution). As the focus, this article systematically retrospects graph embedding-based recommendation from embedding techniques for bipartite graphs, general graphs and knowledge graphs, and proposes a general design pipeline of that. In addition, after comparing several representative graph embedding-based recommendation models with the most common-used conventional recommendation models on simulations, this article manifests that the conventional models can still overall outperform the graph embedding-based ones in predicting implicit user-item interactions, revealing the comparative weakness of graph embedding-based recommendation in these tasks. To foster future research, this article proposes constructive suggestions on making a trade-off between graph embedding-based recommendation and conventional recommendation in different tasks, and puts forward some open questions.

Keywords: Information Retrieval; Recommender Systems; Graph Embedding; Machine Learning; Knowledge Graphs; Graph Neural Networks

Contents

*Corresponding author
Email address: 201921210214@std.uestc.edu.cn (Yue Deng)

1. Introduction

Does big data [1, 2] benefit people's lives? On its face, the question seems absurd. It is true that, for example, the traffic flow big data helps to quantify the potential infectious crowds during the pandemic [3], the scientific research big data could facilitate academic-industry collaboration [4], or the multimedia social big data usually entertains consumers [5]. But meanwhile, the high-volume, high-velocity and high-variety, also called the three "V" features [6], of big data bring problems. Information overload [7, 8] is a case in point, referring to the excess of big data available to a person when making a decision, say, which articles are relevant to a researcher's focus, which products meet a consumer's demand, or which movies pique an audience's interest. Consequently, it would discount one's information retrieval [9] efficiency. Counterbalancing these pros and cons of big data to maximize its benefits requires the development of big data mining techniques [10], among which recommender systems [11–13] have turned out to be a pivotal tool to alleviate the information overload problem, aiming to predict a user's (*e.g.*, researcher, consumer or audience) preferred items (*e.g.*, articles, products or movies) from millions of candidates. Apart from this, recommender systems have seen commercial practices ranging from startup-investors matching [14] to energy efficiency in buildings [15].

Developing recommender systems requires surmounting the sparsity problem [16, 17] and cold start problem [18–21] encountered by recommendation models, the core component of recommender systems. The rationale for recommendation models lies in the accurate inference for user's preferences for items, the prerequisite for well recommendation performance, by analyzing observed user-item relations, among which user-item interactions (Sec. 2.1.1 gives details) are primary resources. However, user-item interactions are usually sparse as a result of only a few of the total number of items that were interacted by a user, called the sparsity problem. When coming to a new user, that no interaction between the user and items yet has been observed leads to the cold start problem, and the same is true of a new item analogically. Given these problems, inadequate user-item interactions as input for recommendation models will lower the accuracy of inference for user's preferences and eventually weaken recommendation performance. As for tackling the sparsity and cold start problems, employing side information [22, 23] or knowledge [24–26] (Sec. 2.1.1 gives details) as a supplement to user-item interactions has been proven promising recently, aiming to uncover hidden (indirect) user-item relations to enrich observed information for recommendation models.

Concerning the ability to efficiently employ side information or knowledge to promote recommendation performance, the discussion about whether graph embedding-based recommendation (Sec. 2.2.1 gives details) usually outperforms conventional recommendation implemented based on graph topological analysis (Sec. 2.1.4 gives details) is an ongoing controversy. With regard to the scalability [16, 27], graph embedding-based recommendation outperforms conventional recommendation, which can efficiently implement recommendation per second for millions of users and items when data is highly complex and large-scale as a result of the three "V" features of side information and knowledge inherited from big data. This result is brought from the two's distinctive rationales: after organizing data (or information) into graph representations (Sec. 2.1.2 gives details), conventional recommendation runs by analyzing a graph's topological characteristics such as users' co-interactions with common items [28] or global topological diffusion [29, 30]. In contrast, graph embedding-based recommendation runs by using nodal embedding vectors, which preserve graph topological features once learned from the graph representations by embedding techniques [31] (Secs. 4.1 and 5.1 give retrospects). In view of that, when employing side information or knowledge, graph embedding-based recommendation can directly reuse the learned nodal embedding vectors rather than repeating the analysis of graph topological characteristics as conventional recommendation does. Therefore, the scalability of it can be substantially improved. Besides, the storability of embedding vectors can support downstream machine learning tasks [32], which require feature vectors of data instances as inputs, like classification [33–39], link prediction [40–43] or clustering [44]. Such a property of embedding vectors enables graph embedding-based recommendation to outperform conventional recommendation in terms of model extensibility.

Nevertheless, as for model explainability (or interpretability) [45]: why did models return such recommendations to a user, graph embedding-based recommendation substantially underperforms conventional recommendation as a result of its general adoption of machine learning methodology [46], almost a black box, whose idea lies on the input-output data fitting for underlying pattern discovery by numerical or analytic optimization methods [47], whereas conventional recommendation can directly realize the explainability through resolving the graph topological characteristics pertaining to a user-item node pair. Although some recent studies argued that by employing knowledge in recommendation[45, 48–50] (Sec. 5.2 gives details) or by causal learning (causal inference) [51–57] to reason and understand user's preferences the explainability of recommendation results can also be indirectly realized, the explainability of recommendation models still faces fundamental limits. In addition, controversies over graph embedding-based recommendation and conventional recommendation are also embodied in recommendation

accuracy. To be sure, by employing side information and knowledge, graph embedding-based recommendation can achieve distinctive improvement in recommendation accuracy beyond conventional recommendation [58–61]. However, this has been cast into doubt by its comparative weakness in some recommendation tasks for predicting implicit user-item interactions compared with conventional recommendation, proved in Sec. 6 on simulations. Similar results were unraveled by Dacrema et al. [62] too.

Faced with these ongoing controversies, the current lack of unified evaluation criterion on graph embedding-based recommendation and conventional recommendation will lead to longstanding discussions in the future, involving extended perspectives from accuracy, scalability, extensibility and explainability, and participated by interdisciplinary researchers ranging from mathematicians to data scientists. In fact, developing both graph embedding-based recommendation and conventional recommendation is not contradictory. The methods of analyzing graph topological characteristics behind conventional recommendation can inspire graph embedding-based recommendation in the utilization of such as subgraphs [63], motifs [64–66], and neighborhood [67–69] to promote its explainability [39]. On the other hand, graph embedding-based recommendation has pioneered novel recommendation scenarios, like conversational recommender system (CRS) [70] or news recommendation [71], providing more promising application prospects for conventional recommendation. It seems that developing both of them to complement each other could improve recommender systems larger than only focusing on one side.

Unlike all-around review articles about conventional recommendation [11–13], newly published reviews on graph embedding-based recommendation [22, 25, 26, 72–77] generally lack a systematic structure and an in-depth description, which seems to be insufficient for researchers focused on conventional recommendation before or interdisciplinary researchers to apprehend this novel prevalent field. To bridge this gap, this article builds an all-around perspective on recommender systems involving both graph embedding-based and conventional methods, and proposes a general design pipeline of graph embedding-based recommendation. As the focus, this article systematically retrospects graph embedding-based recommendation from embedding techniques for bipartite graphs, general graphs and knowledge graphs. In addition, this article further compares the strengths and weaknesses of representative graph embedding-based recommendation models with those of the most common-used conventional recommendation models, on simulations, in different recommendation tasks, revealing that the conventional recommendation models can outperform the graph embedding-based recommendation models in predicting implicit user-item interactions. By analyzing these experimental results, this article provides constructive suggestions on making a trade-off between graph embedding-based recommendation and conventional recommendation, and proposes some open questions for future research.

The rest of this article is organized as follows. Sec. 2 covers basic definitions of subjects and problems, building an all-around perspective on recommender systems and proposing a general design pipeline of graph embedding-based recommendation. Secs. 3, 4, and 5 retrospect embedding techniques for bipartite graphs, general graphs and knowledge graphs, respectively, and then retrospect the graph embedding-based recommendation models based on them, correspondingly. Tabs. 3 and A9 provide an overview of the reviewed models. Sec. 6 presents the experimental results on evaluating representative graph embedding-based recommendation models and the most common-used conventional recommendation models in different recommendation scenarios with distinctive data scales. After analyzing these experimental results, Sec. 6 provides several constructive trade-off suggestions as well as open questions for future research. Finally, Sec. 7 puts forward some prospects on graph embedding-based recommendation, ranging from current challenges to potential solutions.

2. Preliminaries

After introducing several major controversies between graph embedding-based recommendation and conventional recommendation in Sec. 1, in this section Sec. 2.1 devotes to an all-around perspective of recommender systems and illuminates the rationale behind conventional recommendation. Then, Sec. 2.2 illustrates what is graph embedding as well as what is the rationale for it to be applied in recommendation, preparing for Secs. 3, 4 and 5. Following that, Sec. 2.3 proposes a general design pipeline of graph embedding-based recommendation. Finally, the notations used in this article are presented in Tab. 4 at the end of this section.

2.1. Recommender systems

In general, the target of recommendation is to infer user's preferences for items by analyzing user-item relations with observed information (or data) related to users and items, aiming to predict unobserved (or never happened) user-item interactions. This section divides the observed information into three categories: user-item interactions, side information and knowledge, according to their respective distinguishable complexity. Before being employed in

recommendation, the three kinds of information should be represented by graph representations, including bipartite graphs, general graphs and knowledge graphs, correspondingly, which are the bedrock of measuring the k-order proximity between users and items in order to predict unobserved user-item interactions. To clarify the above process, Sec. 2.1.4 takes two common-used conventional recommendation models as instances for illustration.

2.1.1. Information (or data) for recommendation

In engineering, observed information in recommender systems can be recorded by tuples. For example, consider an event that *a 24-year-old male student named Tom watched Iron Man on Netflix on January 28, 2021, and rated this movie with five points*, also called observed information. In engineering, it can be recorded by a tuple like {Tom, male, 24, student, watched, Iron Man, 5, 2021-1-28, Netflix}, in which user-item interactions (*i.e.*, {Tom, watched, Iron Man, 5, 2021-1-28}), side information (*i.e.*, {Tom, male, 24, student}) and knowledge (*i.e.*, {Iron Man, Netflix}) can be further split out. Tab. 1 briefly compares the three kinds of information.

Table 1: A brief comparison between user-item interactions, side information and knowledge. Taking side information or knowledge as a supplement to user-item interactions contributes to a higher recommendation accuracy [58–61], where richer user-item relations are uncovered as the ground of preference inference, which helps to alleviate the sparsity and cold start problems.

Information	Instances	Functions	Comparative complexity
user-item interactions	*e.g.*, user's clicks, browses, or ratings on items	can reflect user's preferences for items	low
side information	*e.g.*, users' social relationships and locations or item's profiles	provides diverse properties of users and items	middle
knowledge	*e.g.* encyclopedias of items	provides abundant direct or indirect relations between items	high

As the primary resources for recommendation, **user-item interactions** can be divided into two categories: explicit ones and implicit ones, according to whether the interactions explicitly carry user's affection degree on items or not. Specifically, **explicit user-item interactions** can be defined as the ratings of items given by users, used to quantify user's affection degree on items based on the assumption that one tends to rate higher on items those he prefers than those he may show indifference. Under this definition, user's rating biases termed **user biases** [78–80] can be dug out from explicit user-item interactions. For example, consider two users: when calculated on a five-point scale, one is used to rate by at least three points on items which he even showed indifference while the other is so extremely strict that never rated by more than three points on items which he even loved. Consequently, the rating biases between the two users obviously exist. In the same way, **item biases** [78, 81] could be brought from the user biases. For example, when it comes to one specific item, its average ratings given by tolerant users or critical users could be different. In this regard, in order to remove the distorted view of user's preferences or item's popularity, taking both the user biases and item biases into account can promote a high-quality recommendation [78, 79]. However, despite their advantages in being able to directly reflect user's preferences for items, the limitation of explicit user-item interactions are clear for the two reasons: (1) In practice, since when surfing online one would prefer to browse, click or watch than rate, accessible explicit user-item interactions are usually sparse, which are insufficient to be as the input of models, not to mention an accurate recommendation. (2) As playing an enhanced role for user's privacy protection [82, 83] in data security, explicit user-item interactions even could be inaccessible in some recommendation scenarios. To tackle these issues, **implicit user-item interactions** have become additional resources, which are defined as a binary state using value 1 to indicate the existence of a user-item interaction (such as click, browse or watch) and value 0 otherwise. Compared to the explicit ones, implicit user-item interactions don't require user's extra operations like ratings or comments on items; so they generally occur more frequently and can be more easily accessed. However, a minor criticism of implicit user-item interactions is that they can't directly carry user's preferences for items, bringing the so-called one-class problem [84], which could resort to converting implicit user-item interactions to explicit ones [85, 86] as a strategy.

In general, user-item interactions are occurring constantly, not merely at a specific time or in a constrained period. Correspondingly, inferred from newly occurring interactions, user's preferences for new items could sprout and those for old items could fade, which means that user's preferences for items could vary over time, usually in long-term and short-term termed as user's **long-term preferences** and **short-term preferences**, respectively. Specifically, user's long-term preferences could vary over time in ways large and small, like the changes of user's personal hobbies (one may prefer comedy movies in his childhood while finding science fiction movies more interesting after entering

college), special events like seasonal changes and holidays or even the changes of one's family status. At the same time, user's short-term preferences could be affected by one's latest interacted items. For example, one's interest in comedy movies may decline after watching lots of comedy movies within a short period. In view of that, capturing both user's long-term and short-term preferences for items contributes to a higher recommendation accuracy [87–89]. For that purpose, by utilizing deep learning methods [90] to mine the underlying patterns involved in user behavior, sequential recommendation [91] has been the recent focus of research into employing user's short-term preferences for items in recommendation. Others include the methods of matrix factorization and Markov processes (Sec. 3.2 gives details). Since not all the reviewed recommendation models in this article take the changes of user's preferences into account, this article defines terms **temporal user-item interactions** and **static user-item interactions** to distinguish recommendation models considering that changes or not, respectively.

To alleviate the cold start and sparsity problems of recommendation, **side information** [22, 23], which is generally characterized by the properties of users and items, is utilized to uncover more hidden (indirect) user-item relations. Back to the first example of this section, the side information {Tom, male, 24, student} records Tom's properties including his gender, age and occupation. In light of that, it could be uncovered that some indirect relations between *Iron Man* and audiences who are close to Tom in these personal properties might exist. In practice, side information usually refers to user's social information [92] and locations [93] or item's profiles [94], labels [95, 96] and textual content [97], to name a few. Among them, as for user's social information, microblogging [98] is a primary resource, which uncovers user's interpersonal relationships (like following or friends) by tweets on social platforms or user's individual profiles. Side information of microblogging has two virtues dear to researchers: abundant and almost real-time [99]. In detail, the abundant information can carry multitudes of diverse relationships between users as well as user's preferences for things that were directly expressed. Such abundance is far beyond that of user-item interactions, which definitely contributes to inferring user's references for items more accurately by, for example, taking one's friends' preferences as a reference. Besides, side information of microblogging is usually considered real-time, benefits from user's tweeting habits that one is inclined to share his daily events or feelings in microblogging or other social platforms. In this way, it provides more opportunities to trace user's latest preferences, which user-item interactions couldn't be that real-time because user's latest preferences can be reflected by these interactions only when new ones occur.

As the most complex one, **knowledge** [24–26] can be displayed or expressed in a language form logically organized by subjects, predicates and objects, related to objective facts of the world. Among the three elements, subjects and objects are usually termed as **entities**, abstract or concrete things of fiction or reality with specific types or attributes. The connections (*i.e.*, predicates) between entities are usually termed as **relations**. With these terms, knowledge can be defined as a collection of entities with different types or attributes and the relations between them. Take the first example of this section one step further, suppose some knowledge about the movie *Iron Man* was excerpted from Wikipedia, displaying that *Iron Man is a 2008 American superhero film based on the Marvel Comics character of the same name. Produced by Marvel Studios and distributed by Paramount Pictures, it is the first film in the Marvel Cinematic Universe (MCU).* Based on that, entities can be split out, like from the fact that *the movie Iron Man belonging to the class of American superhero film the movie* the movie *Iron Man* can be split out as an entity with an attribute of *the first film in the MCU*. Meanwhile, the relation *produced* between the entities *Marvel Studios* and *Iron Man* as well as the relation *distributed* between the entities *Paramount Pictures* and *Iron Man* also can be split out. By employing this knowledge represented by entities and relations in recommendation, hidden (or indirect) relations between Tom and other movies could be uncovered, like that Tom might be inclined to the productions by *Paramount Pictures*, which reflect Tom's underlying preferences for other movies. In practice, the category information on e-commerce platforms, which is organized in a form by tree logic, is also a common-used resource of knowledge, providing a better understanding of user's preferences for items on multiple levels.

To be sure, when speaking of knowledge, one is tempted to argue that, throughout much of recent research, common scientific sense seemed to dictate that knowledge is supposed to be a subcategory of side information, commonly being as a supplement to user-item interactions. Despite general acknowledgement of that, the accepted classification belied the essential distinctions between side information and knowledge by making specious continuity of them. In the first place, the resources of side information and knowledge are different. Side information is usually requested from one's personal information forwardly. For example, when starting to use an APP installed on a cell phone, one might be requested to share his personal information with the APP, at best his name or gender, and at worst his address book or geographical location, which are private. Besides, as for side information of item's profiles, user descriptions of items about such as one's usage experiences are also requested from consumers forwardly. In sharp contrast, since always existed in the real world, knowledge can be naturally perceived and be

displayed or expressed in various forms like encyclopedias or papers. Second, the complexity of side information and knowledge is distinctive, for most of side information is only about the properties of users or items while knowledge is about things more versatile, almost everything in the real world like multi-modal information [100]. In addition, knowledge generally grows constantly and rapidly, making itself far more complex in semantics and multiplicity compared to side information. Third, knowledge is re-usable while side information seldom does the same because the properties of users and items usually change over time. All in all, the three distinctions between side information and knowledge make the techniques employing knowledge in recommendation more challenging than those for side information as a result of the large-scale, multiplicity and evolution characteristics of knowledge beyond side information (Sec. 5.1 gives details).

2.1.2. Graph representations

In the last section, observed information for recommendation is divided into three categories: user-item interactions, side information and knowledge, according to their respective distinguishable complexity. Before being employed in recommendation, the three kinds of information should be represented by graph representations including bipartite graphs, general graphs and knowledge graphs, correspondingly, which are model-readable forms. Tab. 2 briefly compares the three kinds of graph representations. Without loss of generality, this article uses $\mathcal{G} = (E, R, \mathcal{E}, \mathcal{R})$ to denote a graph representation, where E denotes a node-set and R denotes an edge-set. $\mathcal{E}$ denotes a type-set of different nodes and $\mathcal{R}$ denotes a type-set of different edges.

Table 2: A brief comparison between bipartite graph, general graph and knowledge graph. $|E|$ and $|R|$ represent the number of different node types and different edge types, respectively.

Graph representation	Carried information	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	Relative complexity
bipartite graph	user-item interactions	$= 2$	$= 1$	low
general graph	side information	≥ 1	≥ 1	middle
knowledge graph	knowledge	$\to \infty$	$\to \infty$	high

As the most common-used graph representation for user-item interactions, a **bipartite graph** is formally defined as $\mathcal{G}_{bi} = (E, R, \mathcal{E}, \mathcal{R})$ containing two types of nodes and one type of edges (*i.e.*, $|\mathcal{E}| = 2$ and $|\mathcal{R}| = 1$), where edges only exist between nodes with different types. In practice, a bipartite graph can represent users and items in recommender systems with the two types of nodes. Based on that, it can represent implicit user-item interactions by adding edges between the corresponding node pairs and represent explicit user-item interactions by weighting the corresponding edges with ratings. Meanwhile, its use soon widened to algebra. By indexing user nodes and item nodes with rows and columns of a matrix, respectively, a bipartite graph can be directly converted into a matrix, where its elements indicate the existence (*i.e.*, 0/1) or weight (*i.e.*, ratings) of edges in the bipartite graph, which enables the implementation of algebraic theory in recommendation (Secs. 3.1.1 and 3.2.1 give details). Fig. 1 gives toy examples to illuminate how to represent explicit user-item interactions by a bipartite graph and meanwhile to convert it into a matrix.

A **homogeneous graph** [101] $\mathcal{G}_{\text{homo}} = (E, R, \mathcal{E}, \mathcal{R})$ where $|\mathcal{E}| = 1$ and $|\mathcal{R}| = 1$ or a **heterogeneous graph** [102, 103] $\mathcal{G}_{\text{hete}} = (E, R, \mathcal{E}, \mathcal{R})$ where $|\mathcal{E}| > 1$ and/or $|\mathcal{R}| > 1$ can be used to represent side information. Without loss of generality, this article terms the **general graph** to unify both of them. Under this definition, one could consider bipartite graphs as a subcategory of general graphs. However, a bipartite graph would prefer to represent side information but are constraind from doing so by that its edges can only exist between nodes with different types. Since side information could be both homogeneous and heterogeneous, edges should be allowed to exist between nodes of the same type. In contrast, a general graph is more flexible in representing side information, like representing user's social information containing one node type (*i.e.*, user) and one edge type (*i.e.*, friend relationship) by a homogeneous graph, or representing more enriched user's social information with attributes of users and items by a heterogeneous graph. To employ side information in recommendation, it's a common-used way to integrate a general graph with a bipartite graph through their jointly owned nodes as connections (because side information is generally about users and items, which are involved in user-item interactions).

By using triplets (subject, predicate, object) to transform knowledge into a model-readable form, a **knowledge graph** [24] can be used to represent the entities and their relations split out from knowledge. Formally, let E denote the set of subjects and objects and let R denote the set of predicates, knowledge can be represented by triplets $S = (h, r, t)$, where $h \in E$ is termed **head entity** (*i.e.*, subject), $t \in E$ is termed **tail entity** (*i.e.*, object), and $r \in R$ is a directed edge from h to t. Based on these triplets, a knowledge graph can be constructed. Fig. 2

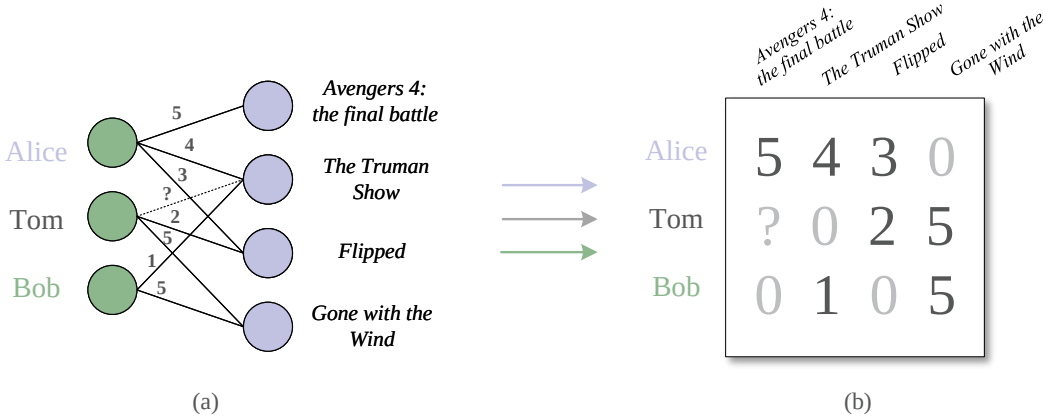

Figure 1: Represent user-item interactions by a bipartite graph and a matrix, respectively. In (a), the weighted bipartite graph represents the explicit user-item interactions from a movie recommender system, where, for example, the edge weighted 5 between Alice and *Avengers 4: the final battle* represents an explicit interaction {Alice, Avengers 4: the final battle, 5}, meaning that Alice watched *Avengers 4* and rated it five points. The bipartite graph in (a) can be directly converted into a weight matrix as shown in (b), where the value 0 corresponds to the unobserved edges in (a), indicating the unobserved interactions between users and movies.

gives toy examples to illuminate the construction process. Since the properties like boundless scale and complex semantics of knowledge, a knowledge graph $\mathcal{G}_{kg} = (E, R, \mathcal{E}, \mathcal{R})$ generally contains extremely diverse node types and edge types such that $|\mathcal{E}| \to \infty$ and $|\mathcal{R}| \to \infty$, which can be considered as the most complex instance of general graphs. In order to sufficiently and efficiently represent these properties of knowledge graphs, research into **multi-viewed graphs** and **multi-layered graphs** has been a tendency recently (Sec. 5.1 gives details). In practice for commercial applications such as those in IBM [104], Amazon [105] or Alibaba [106], representative knowledge graphs include YAGO [107], WikiTaxonomy [108], DBpedia [109], Wikidata [110] and WebOfConcepts [111], to name a few, constructed by automatic knowledge harvesting techniques [112–114]. These open knowledge graphs provide resources for recommendation. Similar in being employed in recommendation to side information dose, a knowledge graph can be integrated with a bipartite graph through their jointly owned nodes as connections. In the same way, a knowledge graph also can be integrated with a general graph.

2.1.3. K-order proximity

After representing observed information for recommendation by graph representations, it comes to measure the **proximity** [115, 116] between users and items (*i.e.*, between the corresponding user nodes and item nodes in graph representations), which is a key step of recommendation's implementation, from the perspective of link (*i.e.*, edge) prediction [41, 42]. Specifically, user-item proximity is used to record or predict the similarity between user-item node pairs both linked by edges or not, which can be used to quantify the likelihood of occurrence of unobserved user-item interactions in recommender systems. Intuitively, the higher similarity between a not-linked user-item node pair, the higher user's affection degree on the item, and then a higher likelihood of occurrence of the corresponding unobserved user-item interaction.

In general, the **first-order proximity** of a node pair $v_i - v_j$ is defined as their local pairwise proximity, which can be measured by the existence (*i.e.*, 0/1) or weight (*i.e.*, rating if exists) of edge (v_i, v_j). Take Fig. 1 as an example, the first-order proximity of node pair Tom-*Flipped* can be measured as 2 by the edge's weight (*i.e.*, rating), recording the similarity between them. Since the edge between Tom and *Avengers 4: the final battle* did not exist, the similarity between Tom-*Avengers 4: the final battle* need to be predicted based on the observed user-item proximity by methods (*i.e.*, recommendation models). However, edges in a graph representation are usually in a small proportion as a result of the sparsity problem of recommendation, which casts the first-order proximity into doubt about its precision on recording or predicting user-item similarity. In fact, the proximity between two not-linked nodes does not always have to be zero measured by the first-order proximity, like in situations where

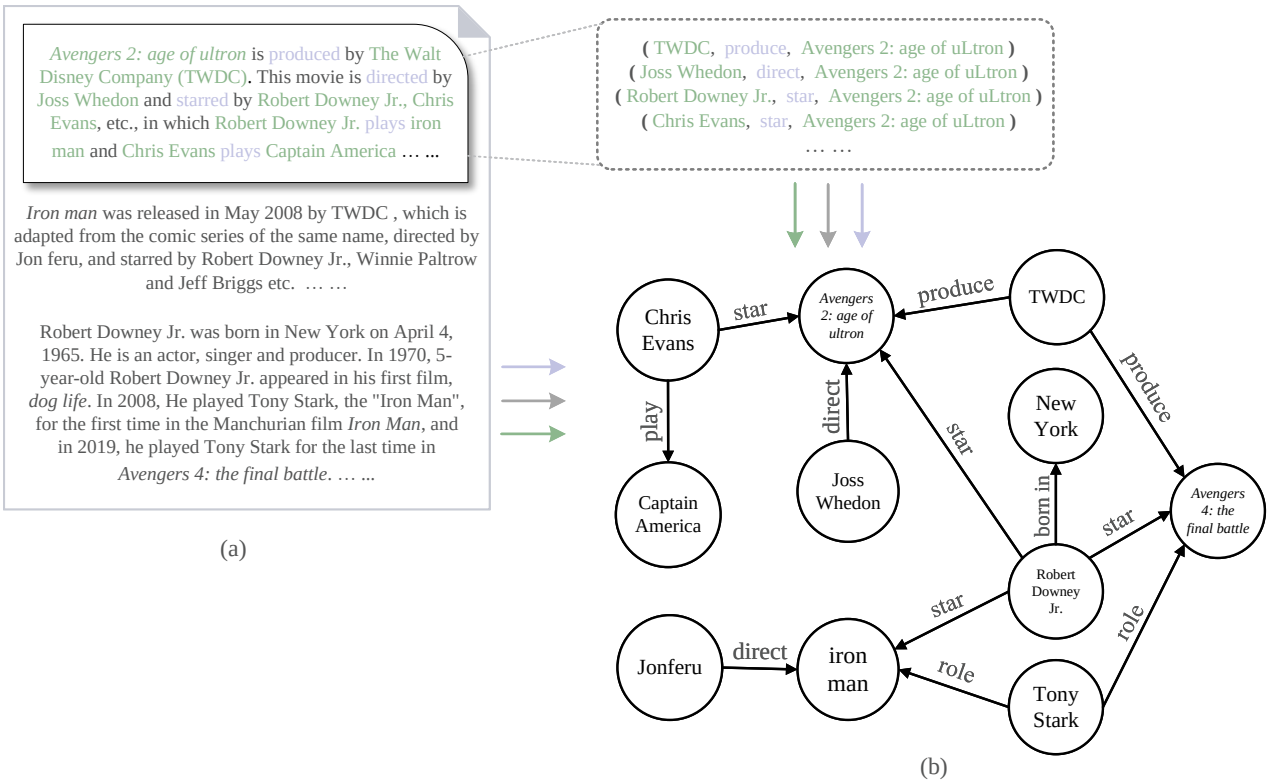

Figure 2: Represent knowledge by a knowledge graph. In (a), the subjects, predicates and objects contained in the profile of a classic movie *Avengers 2: age of ultron* can be split out and can be represented by multiple triples organized as (subject, predicate, object). Then, these triplets are used to construct a knowledge graph shown in (b).

the corresponding two users could still be intrinsically related. For example, the first-order proximity of Tom-Bob in Fig. 1 is measured as zero while the two users might share common movie preferences, which can be intuitively inferred from the fact that they co-rated the movie *Gone with the Wind* with the same points of five, and thus be intrinsically related. To make up for the flaw of the first-order proximity, the **second-order proximity** of a node pair $v_i - v_j$ is defined as the overall first-order proximity between the two nodes' respective neighborhoods. Formally, denote $S_i = \{s_{i,1}^{(1)}, s_{i,2}^{(1)}, ..., s_{i,|V|}^{(1)}\}$ as a collection of the first-order proximity $s_{i,j}^{(1)}$ between node v_i and the other nodes v_j in a graph representation, respectively. Following that, the second-order proximity between v_i and v_j is measured based on S_i and S_j by such as cosine index [117], Pearson coefficient (PC) [118] or Jaccard index [119]. Apparently, if two nodes v_i and v_j do not have any other co-lined node, the second-order proximity of $v_i - v_j$ is measured as zero, such as that of *Avengers 4: the final battle*-*Gone with the Wind* shown in Fig. 1. Iteratively, the **higher-order proximity** of a node pair can be analogously defined as above, which has been a popular research focus in recent years (Sec. 4.1.2 gives details).

Under these definitions of proximity with different orders, it is more obvious why is it feasible to alleviate the cold start and sparsity problems of recommendation by employing side information or knowledge. Take Fig. 1 for example, when coming to a new movie *Avengers 2: age of ultron* represented by a node, it is isolated in the bipartite. Consequently, it is impossible to measure the proximity between the new movie and any other user or movie, let alone to predict the similarity between them and estimate user's preferences on this new movie, which leads to the cold start problem as illustrated beforehand. However, after employing side information or knowledge in recommendation, it is another matter. For example, by integrating the knowledge graph shown in Fig. 2 with the bipartite graph, more hidden (or indirect) relations between the new movie *Avengers 2: age of ultron* and other movies can be uncovered, like that with *Avengers 4: the final battle*. These new relations enable the implementation of measuring user-item proximity related to the new movie. Benefits are not limited to this, enriched relations

uncovered by side information or knowledge can also make the original graph denser, with newly established edges between nodes. In this way, not only the sparse problem of recommendation could be alleviated but also user-item proximity could be put in more diverse orders, ensuring the precision of measured user-item similarity.

In different recommendation methods, the value of user-item proximity is generally entitled different practical meanings, like the ratings of items given by users by matrix factorization-based method (Sec. 3.1.1 gives details), the Pearson coefficient (PC) [118] between nodes by k-nearest neighbors-based method (Sec. 2.1.4 gives details), the allocated resources on items diffused from users by diffusion-based method (Sec. 2.1.4 gives details) or the occurrence probability of user-item interactions by deep learning-based method (Sec. 3.1.3 gives details). On the other hand, the value of user-item proximity could be meaningless, such as that in translation-based method (Sec. 4.1.1 gives details). Without loss of generality, this article generalizes the definition of proximity to be a metric, which can be used to quantify the relative magnitude of similarity between nodes in graph representations. Note that the term proximity also can be called similarity.

2.1.4. Methods and conventional models

As illustrated above, in order to quantify the likelihood of occurrence of unobserved user-item interactions, recommendation models aim to predict the similarity between unobserved user-item pairs based on observed user-item proximity, among which **collaborative filtering** [28, 120–123], **diffusion-based** [14, 30, 124, 125] and **content-based** [126–128] are three prevalent ones. Respectively, based on the assumption that a user's preferences for items might be affected by those of his neighbors, collaborative filtering method predict the similarity between an unobserved user-item pair by analyzing the observed proximity between the item and the user's neighbors sharing interacted items with the user. Diffusion-based method pioneered a strategy for applying physic diffusion processes, such as heat spreading [29] or mass diffusion [30], to recommendation. Content-based method runs by building user's profiles, which are used to match with item's attributes or descriptions. As for conventional recommendation models, user-based collaborative filtering (UBCF) [28] and probabilistic spreading (ProbS) [30] are two common-used ones of collaborative filtering method and diffusion-based method, respectively. The rest of this section illustrates the two models, used as instances to illuminate the rationale behind conventional recommendation implemented by directly analyzing graph topological characteristics. At the same time, the two models are used as experimental benchmarks in Sec. 6.

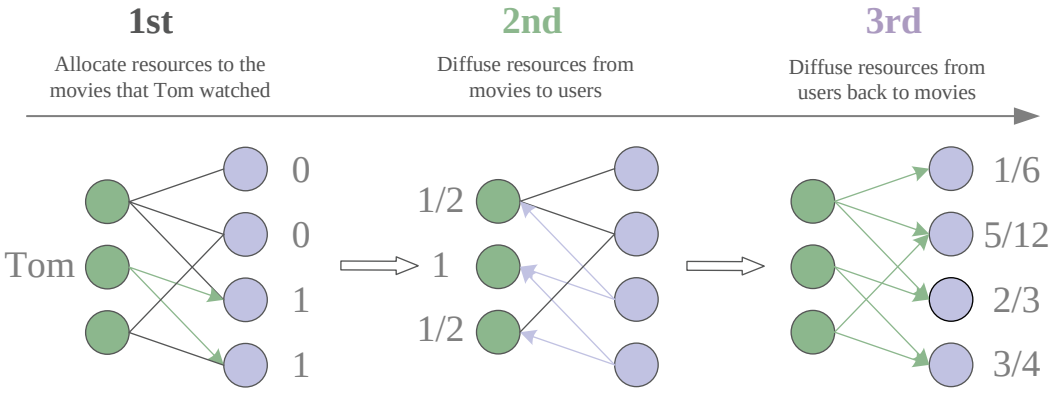

Figure 3: Schematics of ProbS. The first step is to allocate resources. The second and third steps combine a two-step process of resource diffusion.

Based on the bipartite graph shown in Fig. 1, to predict Tom's similarity with the two movies not interacted with him, UBCF measures the second-order proximity between Tom and other users by Pearson coefficient (PC) [118] as

$$S_{i_1 i_2}^{\mathrm{PC}} = \frac{\sum_{j\in\mathcal{R}(i_1)^+\cap\mathcal{R}(i_2)^+}(r_{i_1 j}-\overline{r}_{i_1})(r_{i_2 j}-\overline{r}_{i_2})}{\sqrt{\sum_{j\in\mathcal{R}(i_1)^+\cap\mathcal{R}(i_2)^+}(r_{i_1 j}-\overline{r}_{i_1})^2}\sqrt{\sum_{j\in\mathcal{R}(i_1)^+\cap\mathcal{R}(i_2)^+}(r_{i_2 j}-\overline{r}_{i_2})^2}}, \tag{2.1}$$

where $\mathcal{R}(i_1)^+ \cap \mathcal{R}(i_2)^+$ denotes the set of items rated by both users i_1 and i_2, and $\overline{r}_i$ denotes the average of ratings given by user i. Following that, the similarity between Tom and *The Truman Show*, which represents the predicted rating $\hat{r}_{*j}$ of *The Truman Show* (denoted by j) given by Tom (denoted by $*$), can be calculated by

$$\hat{r}_{*j} = \overline{r}_* + \alpha \sum_{i \in N(*)} S_{*i}(r_{ij} - \overline{r}_i), \tag{2.2}$$

where $N(*)$ is a collection of neighbors within the second-order hops from Tom, and $\alpha = \dfrac{1}{\sum_{i \in N(*)} |S_{*i}|}$ is a normalization factor. For example, in Fig. 1, based on the observed proximity that Alice and Tom both rated *Flipped* as well as Bob and Tom both rated *Gone with the Wind*, and calculated by Pearson coefficient in Eq. (2.1), the second-order proximity between Tom and Alice can be measured as -0.9795 and that between Tom and Bob can be measured as 0.8593. Following that, based on the observed proximity that Alice rated *The Truman Show* by 4 points and Bob rated *The Truman Show* by 1 point, the similarity $\hat{r}_{*j}$ (*i.e.*, predicted rating) can be calculated by $3.5 + \dfrac{1}{1.8388}[-0.9795 \times (4-4) + 0.8593 \times (1-3)] \approx 2.5654$, according to Eq. (2.2).

Different from UBCF resolving topological characteristics of users' co-interactions with common items, ProbS runs a mass diffusion process on a graph, using dynamically diffused and aggregated resources to represent the similarity between nodes. Take Tom in Fig. 3 as the user waiting to be recommended, in the first step, ProbS allocates a unit of resources for all the movies that interacted with Tom, respectively. Then, in the second step, the resources allocated to movies are equally distributed and diffused along edges from each movie to its interacted users, where the aggregated resources reached to Alice and Bob can be used to represent their respective second-order proximity with Tom. In the third step, these resources allocated to users are again equally distributed and diffused along edges from each user back to his interacted movies. Finally, the aggregated resources reached to movies can be used to represent their similarity with Tom, which indicates that Tom's preference for *The Truman Show* could be beyond that for *Avengers 4: the final battle*. Note that the second and third steps of ProbS combine a two-step diffusion process, which can be iterated to multiple rounds for a higher recommendation accuracy.

2.2. Graph embedding

In the last section, the rationale for conventional recommendation is illuminated by two models as instances, which are implemented by resolving graph's topological characteristics. However, this rationale prohibits conventional recommendation from efficiently implementing on different recommendation scenarios, especially with big data, because it has to repeat the resolution on graph representations mechanically, which lags behind graph embedding-based recommendation in scalability and migration. In contrast, graph embedding-based recommendation runs by directly reusing nodal embedding vectors, which represent users and items, once learned from graph representations. On top of that, after incorporating machine learning methodology, graph embedding-based recommendation can be equipped with abilities to pattern discovery, which contributes to a higher recommendation accuracy. This section presents basic concepts of graph embedding as well as its application in recommendation.

2.2.1. Definitions and concepts

Graph embedding [31, 129–131] is a technique used to generate features of non-Euclidean data for machine learning-based downstream tasks, like node classification [33, 34], graph classification [35–39], link prediction [40–43], clustering [44] and stuff. In general, in order to satisfy the input format of machine learning models [132], features (usually represented by multidimensional vectors) of objects in data should be generated in the first place. For that purpose, until recently, researchers usually resorted to artificial generation methods [133] implemented based on hand-engineering with expert knowledge and bag-of-words methods [134] as strategies for generating features of Euclidean data. However, as illustrated in Secs. 2.1.1 and 2.1.2 that most of the information (or data) for recommendation is represented by graph representations with complex and diverse hidden relation (or connectivity) patterns, characterized by non-Euclidean. For that gap, in recent research, graph embedding techniques began, for the first time in large numbers, to be used to generate features of non-Euclidean data, which can project non-Euclidean graph representations into a low-dimensional Euclidean space consisting of embedding vectors (also called **embeddings**) of nodes, edges, subgraphs or whole graphs as their features. These embeddings preserve the intrinsic topological characteristics of graph representations, which can be used to reconstruct them.

Formally, in terms of generating features of nodes, graph embedding is defined as a mathematical process using a mapping $\Phi : \mathcal{G} \to \mathbb{R}^{n \times k}$ to project a graph representation $\mathcal{G}$ into a Euclidean space $\mathbb{R}^{n \times k}$, where n is the size of $\mathcal{G}$ (*i.e.*, $\mathcal{G}$ contains n nodes) and k ($k \ll n$) is the Euclidean space's dimension. Through the mapping, the embedding

(a k-dimensional vector) of an arbitrary node v_i in $\mathcal{G}$ can be generated as $\Phi(v_i) \in \mathbb{R}^k$. Based on these embeddings, the proximity between two nodes v_i and v_j can be measured as $\mathcal{F}(\Phi(v_i), \Phi(v_j))$, where $\mathcal{F} : (\cdot, \cdot) \to \mathbb{R}$ (like dot-product [135]) is a mapping that can project two embeddings to $\mathbb{R}$, where a greater value of $\mathcal{F}(\Phi(v_i), \Phi(v_j))$ is generally recognized as a higher possibility of the existence of edge (v_i, v_j). Following that, $\mathcal{G}$ can be approximately reconstructed by adding these possible edges between node pairs. Intuitively, a well-performed graph embedding technique can determine a mapping $\Phi : \mathcal{G} \to \mathbb{R}^{n \times k}$ related to $\mathcal{F} : (\cdot, \cdot) \to \mathbb{R}$, whether it is directly set up or learned by machine learning methods, which is supposed to be used to approximately reconstruct $\mathcal{G}$'s topological characteristics as much as possible.

When it comes to machine learning-based graph embedding techniques, which are the most prevalent ones in recent years [132], the process of determining the mapping $\Phi : \mathcal{G} \to \mathbb{R}^{n \times k}$ runs by optimization algorithms (Sec. 2.2.3 gives details). Specifically, given a graph representation $\mathcal{G}$ with a node-set V, constructing **training samples** and **test samples** is the first step. With a set of node pairs $S^{\text{train}} = \{(v_i^{\text{train}}, v_j^{\text{train}}) | v_i^{\text{train}}, v_j^{\text{train}} \in V\}$ randomly sampled from $\mathcal{G}$ by a specified proportion and a set of their observed proximity $P^{\text{train}} = \{p_{ij}^{\text{train}} | p_{ij}^{\text{train}} \in \mathbb{R}\}$, correspondingly, training samples can be constructed as $(S^{\text{train}}, P^{\text{test}})$, and in the same way, so do test samples $(S^{\text{test}}, P^{\text{test}})$ that satisfy $S^{\text{train}} \cap S^{\text{test}} = \emptyset$. Then, in the second step, from a beforehand defined hypothesis space Φ [132] containing all possible mappings, machine learning-based graph embedding techniques aim to learn a candidate mapping $\Phi^{\text{local}} \in \Phi$ that could minimize the average error between predicted proximity $\mathcal{F}(\Phi^{\text{local}}(v_i^{\text{train}}), \Phi^{\text{local}}(v_j^{\text{train}}))$ and observed proximity p_{ij}^{train} on training samples. Next, in order to assess their performance, the average error between $\mathcal{F}(\Phi^{\text{local}}(v_i^{\text{test}}), \Phi^{\text{local}}(v_j^{\text{test}}))$ and p_{ij}^{test} on test samples is calculated. Until the error is lower than a target precision, the learning process on training samples will iterate run by optimization algorithms, searching for the optimal $\Phi^{\text{global}} \in \Phi$ that can minimize that average error on test samples. In fact, the rationale behind machine learning-based graph embedding lies in the (high-order) input-output data fitting, aiming to learn embeddings that can capture and preserve the complex patterns hidden in graph representations as much as possible, which contributes to reconstructing the original graphs. Unless otherwise specified, the graph embedding techniques retrospected in this article are all machine learning-based.

2.2.2. Recommendation based on graph embedding

When incorporating graph embedding techniques in recommendation, it comes to a sort of straightforward. By implementing graph embedding techniques on graph representations involving user nodes and item nodes, the embeddings of users and items can be learned to measure user-item proximity and predict their similarity for recommendation, which is called **graph embedding-based recommendation**. Methodologically, similar in the process of learning a mapping $\Phi : \mathcal{G} \to \mathbb{R}^{n \times k}$ to graph embedding techniques, in the first place graph embedding-based recommendation constructs two hypothesis spaces U and V for users and items, respectively, from which two mappings that project user nodes and item nodes of $\mathcal{G}$ into a common Euclidean space can be determined. In order to determine the optimal mappings, **objective functions** are set up to measure the average error between predicted proximity $\mathcal{F}(U(v_i), V(v_j))$ of user-item node pairs and observed ones p_{ij} on both training samples and test samples, formally constructed as $E[L(\mathcal{F}(U(v_i), V(v_j)), p_{ij})]$ where L is a **loss function** and E is an objective (or expectation) function. Based on it, the embeddings of users and items can be learned by optimization algorithms. Using these embeddings, the probabilities of existence of each unobserved user-item interaction can be predicted by $\mathcal{F}(U(v_i), V(v_j))$, which is the ground for sorting candidate items in descending order and select the top-N ones as recommendations returned to users.

From a practical perspective, the embeddings of users and items learned from side information and knowledge related to users and items are reusable and are possibly optimal for preserving and reconstructing the original graph representations. In that case, they could intrinsically carry the properties of users and items as well as the hidden relations between users (since the embeddings of edges can be derived from the embeddings of their endpoints through some calculations). Resorting to combing these embeddings as a supplement with those learned from user-item interactions, enriched information can be employed in recommendation, which contributes to alleviating the cold start problem by building relations between ever non-interacted user-item pairs and the sparsity problem by uncovering more hidden relations between sparsely connected user-item pairs. As an overview, categorized by different recommendation tasks, the graph embedding-based recommendation methods retrospected in this article are summarized in Tab. 3, including their respective pros, cons and recent focus.

In practice, the applications of graph embedding techniques are not only throughout recommender systems but far outside it as well [77], like knowledge graph completion, question answering and query expansion.

Table 3: A comparison between different graph embedding-based recommendation methods. Categorized by recommendation tasks, recent focuses concentrate on developing the pros and solving the cons of the respective methods.

Task	Method	Section	Pros	Cons	Recent focus
Bipartite graph embedding for recommendation	Matrix factorization (MF)	3.1.1 3.1.4 3.2.1	1. has well extensibility 2. can achieve non-negative embedding 3. can capture user's long-term preferences	1. faces non-convex optimization problem 2. is shallow learning 3. could violate the triangle inequality principle	1. non-negative MF 2. metric learning 3. fast online learning
	Bayesian analysis	3.1.2	1. can achieve automatic hyperparameter adjustment 2. can achieve pairwise ranking	1. is shallow learning	1. automatic machine learning 2. casual inference
	Markov processes	3.2.2	1. can capture user's short-term preferences	1. is shallow learning	1. combined with MF
	Deep learning	3.1.3	1. can discover nonlinear patterns 2. can achieve fast parallel computing 3. has well input compatibility	1. lacks explainability 2. faces higher hyperparameter adjustment difficulty	1. deep metric learning 2. casual learning 3. sequential recommendation
General graph embedding for recommendation	Translation	4.1.1 4.2	1. can preserve local topological features 2. can flexibly distinguish the multiplicity of nodes and relations	1. could lose global topological features	1. sequential recommendation
	Meta path (Random walk)	4.1.2 4.2	1. can preserve global topological features	1. requires expert knowledge for meta path design	1. random walk on heterogeneous graphs 2. combinations with MF 3. automatic meta path construction by using graph topology
	Deep learning	4.1.3 4.2	1. can preserve nonlinear topological features 2. can run (semi-)unsupervised learning	1. could lose information through the Encoder	1. attention mechanism and self-attention mechanism
Knowledge graph embedding for recommendation	Graph neural network (GNN)	5.1 5.2	1. can achieve fast parallel computing 2. can capture the multiplicity and dynamics of knowledge graphs	1. carries popularity biases in negative sampling	1. propagation module 2. sampling module 3. pooling module
	Multi-viewed graph	5.1 5.2	1. can capture relational multiplicity	1. could lose nodal multiplicity	1. attention mechanism for weight distribution
	Multi-layered graph	5.1 5.2	1. can be used for cross-domain recommendation	1. requires expert knowledge for modeling	1. translation method 2. meta path method 3. attention mechanism

2.2.3. Optimization algorithms

Objective functions, once formulated, can be solved as optimization problems by numerical or analytical **optimization algorithms** [47, 136, 137], which are used to implement the learning process of searching for the optimal U^{global} and V^{global} from beforehand defined hypothesis spaces, in order to satisfy the extremum of objective functions. In this way, the effectiveness and efficiency of optimization algorithms determine the performance of graph embedding-based recommendation.

Briefly, common-used optimization algorithms for graph embedding-based recommendation include stochastic gradient descent (SGD) [138] and its parallel version ASGD [139], the two most popular ones out of their simplicity and efficiency, as well as other representative latest advances like Mini-bath Adagrad [140], nmAPG [141], Adam

[142] and ADMM [143].

2.3. A general design pipeline of graph embedding-based recommendation

Overall, under the framework of machine learning methodology, a graph embedding-based recommendation model can be mathematically presented as

$$\arg\min_{\Phi} E\Big(L\big(\mathcal{F}(\Phi(v_i),\Phi(v_j)),p_{ij}\big),\Theta\Big),\Phi\in H,\forall v_i,v_j\in\mathcal{G}, \tag{2.3}$$

where $\Phi:\mathcal{G}\to\mathbb{R}^{n\times k}$ denotes a mapping that projects a graph representation's n nodes into an embedding matrix $\mathbb{R}^{n\times k}$ consisting of k-dimensional row vectors, H denotes a hypothesis space designed beforehand, $\mathcal{F}:(\cdot,\cdot)\to\mathbb{R}$ denotes a proximity measurement, L denotes a loss function which is used to measure the error between predicted values and observed (or true) values on training samples or test samples, E denotes an objective function measuring the expectation of overall loss, p_{ij} is the observed proximity between nodes i and j, and Θ denotes a hyper-parameter set. From the perspective of Eq. (2.3), this section decomposes the designing process of graph embedding-based recommendation into several steps and proposes a general design pipeline of that, as shown in Fig. 4.

Specifically, in Fig. 4, the first step is to collect information for recommendation from such as public data sets, practical recommender systems, Internet of things or other commercial data products, among which public data sets are recognized as the priority of designing and evaluating a recommendation model because of their cost-effective and widespread access. Then, representing the information by graph representations is the second step. In this step, selecting an appropriate graph representation $\mathcal{G}$ that can capture and preserve the complexity (or multiplicity) involved in original information as much as possible is crucial, which directly determines the design of recommendation models and the accuracy of recommendation results. After that, the third step is to build proximity measurements $\mathcal{F}:(\cdot,\cdot)\to\mathbb{R}$, used to measure the proximity between node pairs in graph representations. Methodologically, as for a specific recommendation scenario, once identified, the first-order proximity could give clue to the possible forms of proximity measurements fitting this scenario, which helps researchers build higher-order ones.

When it comes to designing a graph embedding-based recommendation model (the fourth step in Fig. 4), a hypothesis space H should be constructed in the first place, from which the optimal mapping Φ in Eq. (2.3) can be searched out. On training samples, after determining an initial mapping Φ from a constructed hypothesis space, nodal embeddings can be generated through it. As the input of $\mathcal{F}$, these embeddings can be used to measure the observed proximity between node pairs. In order to assess the precision, loss function L is designed to calculate the error between predicted proximity and the corresponding observed (*i.e.*, true) proximity of a node pair. After being implemented on all node pairs of training samples, the expected loss can be calculated by an objective function E, an expectation function. Searching out (or training) the optimal mapping Φ on training samples from the hypothesis space runs by optimization algorithms, which will be used to predict unobserved proximity on test samples. Methodologically, designing an appropriate loss function, objective function or optimization algorithm is generally not hectic because a lot of related theories and experiences have matured as a recipe, which can be easily accessed from previous research. In fact, what really matters in step 4 is to construct a hypothesis space fitting to a specific recommendation task. Completely pioneering a novel model is usually not easy. In this regard, as references, Tabs. 5, 6, 7, 8, 9, 10 and 11 summarize several common-used architectures of different graph embedding techniques and different recommendation methods, including their respective hypothesis spaces, loss functions or objective functions.

As shown in Fig. 4, the four steps are recurrent as an iteratively revising and refining process. For example, when designing a recommendation model, if its performance has hit its ceiling while still not being able to reach one's expectation, it could be helpful to modify or even reconstruct a more appropriate hypothesis space or proximity measurement. In this process, as said before, constructing appropriate graph representations which can capture and preserve the complexity (or multiplicity) of original information is crucial. For that purpose, combined with analyzing the topology characteristics of a graph representation constructed in step 2, going back to step 1 to check if it can well fit the original information is a strategy, which gives clues to refine the structures of graph representations.

On the other hand, the design pipeline on its face is data-oriented (or task-oriented), which might be preferred by computer science researchers, who generally design recommendation models starting from specific tasks related to collected data (*i.e.*, information). To be sure, by means of data mining techniques, this data-oriented designing strategy could quickly dig out the hidden patterns of data and incorporate them in modeling, which can achieve a higher recommendation accuracy on a specific task. However, these models designed in this way face fundamental

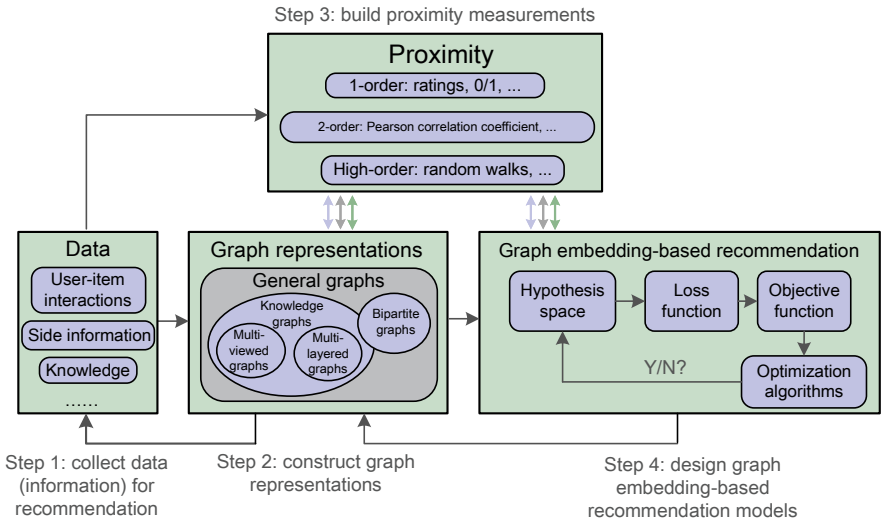

Figure 4: A general design pipeline of graph embedding-based recommendation.

limits on their generalization to other tasks, because they are data-oriented while different tasks generally carry distinctive data patterns, scales or sparsity. On top of that, by starting from step 4, the designing process based on this pipeline can also be run by generalization-oriented, aiming to design versatile recommendation models fitting to diverse tasks with different data properties, in which case physicists and mathematicians might prefer. For all they are two different perspectives for designing models, there is no priority between data-oriented and generalization-oriented strategies. In fact, by combining their respective advantages, it could be more beneficial for researchers to design recommendation models, where multi-task learning [144, 145] seems to be a promising direction.

At the end of this section, notations used in this article are presented in Tab. 4. The following Secs. 3, 4 and 5 retrospect embedding techniques for bipartite graphs, general graphs and knowledge graphs, respectively, as well as their corresponding applications in recommendation.

Table 4: Notations used in this article.

Notations	Meaning
R	The observed user-item rating matrix, where its element r_{ij} represents the rating of item j given by user i.
$\hat{R}$	The predicted user-item rating matrix.
$\mathcal{R}(i)^+, \mathcal{R}(i)^-$	The set of items rated (unrated) by user i.
$N(i)^+, N(i)^-$	The set of items implicitly interacted (non-interacted) by user i.
U	The user embedding matrix consisting of row vectors U_i of each user i.
V	The item embedding matrix consisting of row vectors V_j of each item j.
S	The item-item (user-user) proximity (or similarity) matrix, where $S_{j_1 j_2}$ ($S_{i_1 i_2}$) is the proximity between items j_1 and j_2 (users i_1 and i_2).
b_{ij}	The user-item interaction bias involved in rating r_{ij}, consisting of user bias b_i and item bias b_j.
$\mathcal{G}_{kg} = (E, R, \mathcal{E}, \mathcal{R})$	A knowledge graph, where E is the set of entities and R is the set of relations, $\mathcal{E}$ and $\mathcal{R}$ represent the set of node types and the set of relation types, respectively.
(h, r, t)	A knowledge triplet, where $h, t \in E$ represent a head entity and a tail entity, respectively; $r \in R$ represents the relation between entities.
$\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r}$	The embedding vectors of h, t and r.
$s_{ij}^{(1)}, s_{ij}^{(2)}$	The first-order proximity and second-order proximity between nodes v_i and v_j.

3. Bipartite graph embedding for recommendation

To reveal user's preferences for items, recommendation models are generally run by analyzing user-item relations which are directly recorded in observed user-item interactions and also can be uncovered with side information or knowledge. As the bedrock, recommendation models based on bipartite graphs are of top priority in research, which can be generalized to recommendation models based on general graphs or knowledge graphs. According to the taxonomy of user-item interactions (illustrated in Secs.2.1.1), Secs. 3.1 and 3.2 retrospect recommendation models based on bipartite graph embedding techniques for static user-item interactions and temporal user-item interactions, respectively.

3.1. Recommendation with static user-item interactions

In general, recommendation models based on bipartite graph embedding techniques for static user-item interactions can be divided into three categories: those based on methods of matrix factorization, Bayesian analysis and deep learning. From an overview, as the pioneer of bipartite graph embedding techniques, the matrix factorization method has a virtue of extensibility dear to researchers. As a probabilistic version of the matrix factorization method, the Bayesian analysis method can alleviate the non-convex optimization issue out of data sparsity problem suffered by the matrix factorization method, in a manner that setting model's regularization terms with prior knowledge, like the fact that the error follows a Gaussian distribution. As for learning and preserving the non-linear patterns involved in data, the deep learning method has significant advantages over the above two methods.

3.1.1. Models based on matrix factorization

The rationale behind matrix factorization-based recommendation models basically lies in the singular value decomposition (SVD) [146], which can decompose a matrix $A_{M\times N}$ into $A = U\Sigma V^T$, where U and V are two orthonormal Eigen matrices and Σ is a diagonal matrix composed of A's singular values. In turn, implementing matrix product on U, V and Σ can approximately reconstruct A. Within this framework, through SVD, a user-item rating matrix R is supposed to be decomposed into such elements as the embedding matrices of users and items, based on which R can be approximately reconstructed by implementing matrix product on the embedding matrices as well.

For that purpose, latent semantic analysis (LSA) [147] is recognized as one pioneer of SVD's application in textual information retrieval. Based on documents and terms appearing in at least two documents, LSA firstly constructs a term-document matrix A where its element a_{ij} denotes the frequency of term i's appearance in document j. Then, through truncated SVD [148] (an accelerated version of SVD), A can be decomposed by $A \approx \hat{A} = U_k\Sigma_k V_k^T$, based on which the embedding of term i can be represented by the i-th row of matrix $U_k\Sigma_k$ and that of document j can be represented by the j-th row of matrix $V_k\Sigma_k$, which are both in a common k-dimensional vector space. To complete a user's information retrieval with a query q (a set of query words), LSA can generate the embedding of q as $\hat{q} = q^T U_k\Sigma_k^{-1}$, which will be used to measure the query q's proximity with each of the documents, by doing operations (such as dot product) on their corresponding embeddings.

Feasible as LSA in theory, when coming to recommender systems where the number of users and items are generally hundreds of millions, LSA becomes unfeasible in decomposing such an extremely huge user-item interaction matrix R as a result of the high complexity of SVD and the sparsity of U and V which brings the NP-hard problem [149]. To break those limitations, on his blog Simon Funk proposed FunkSVD inherited from LSA's idea, which resorts to optimization algorithms as a strategy for efficiently running on a large-scale matrix for recommendation (Tab. 5 gives details). Slightly different from LSA, FunkSVD does not directly decompose R by $R = U\Sigma V^T$ but rather hypothesizes that R can be represented by the dot product of two matrices U and V, the embedding matrices of users and items. After initializing their element values, FunkSVD will search the optimal U and V by optimization algorithms, satisfying $UV^T = \hat{R} \approx R$ as approximately as possible.

FunkSVD has several virtues dear to researchers. One remarkable aspect of those is its salient extensibility, which makes it compatible with auxiliary information (such as user biases or item biases) contributing to a higher recommendation accuracy. In view of that, FunkSVD's variants soon widened in subsequent research. For instance, by defining the biases in ratings as a term $b_{ij} = \mu + b_i + b_j$ linearly appended to UV^T, BiasSVD [78] can incorporate user biases b_i and item biases b_j (Tab. 5 gives details) into FunkSVD. By defining user's preferences as a term (Tab. 5 gives details) appended to U_i, SVD++ [78] can further incorporate user's implicit interactions into BiasSVD, which decreases the deviations between b_{ij} and $U_iV_j^T$. Auxiliary information that can be incorporated into recommendation is not limited to these forms. For instance, on the advice of pattern mining and data analysis, Hu et al. [154] discovered that a positive correlation could hide out between an individual business' ratings given by customers

Table 5: Examples of modeling matrix factorization-based recommendation. (1) In FunkSVD, by penalizing the magnitudes of parameters, $\lambda(\|U_i\|_2^2 + \|V_j\|_2^2)$ is used as a regularization term for preventing from overfitting [150]. (2) In BiasSVD, μ is the average of the values in R and b_i (b_j) is the user (item) bias. (3) In SVD++, $|N(i)^+|^{-\frac{1}{2}} \sum_{k \in N(i)^+} y_k$ represents user i's preferences for his implicitly interacted items, where y_k is item k's embedding vector. (4) In SRui, $\mathcal{F}^+(i)$ and $\mathcal{Q}^+(j)$ represent the top-N social neighbors of user i and item j, respectively, whose proximity s_{**} is measured by Pearson correlation coefficient [151]. (5) In NCRPD-MF, v_n and v_c represent the features of item's geography information and category, respectively; and the received review words v_w of an item can be used to represent its intrinsic characteristics with V_j. $\alpha_1, \alpha_2 \in [0, 1]$ correspondingly control the influence of geographical neighborhood and category; and z represents both popularity and geographical distance. (6) In FM, in order to predict the rating $\hat{r}(\boldsymbol{x})$ of an item given by a user, a feature vector $\boldsymbol{x}$ is constructed, consisting of the features of the user and the item both represented by one-hot encoding and the ratings of other items rated by the user and stuff. $w_0, w_i \in \mathbb{R}$ represent the global bias and the strength of the i-th variable, respectively; and $\boldsymbol{v}_i$ represents the embedding of the i-th feature in all n features.

Model	Factorization (hypothesis space)	Objective function
FunkSVD	$\hat{r}_{ij} = U_i V_j^T$	$\underset{U,V}{\arg\min} \sum_{(i,j) \in \mathcal{R}(i)^+} (r_{ij} - U_i V_j^T)^2 + \lambda(\|U_i\|_2^2 + \|V_j\|_2^2)$
BiasSVD	$\hat{r}_{ij} = \mu + b_i + b_j + U_i V_j^T$	$\underset{U,V,b_*}{\arg\min} \sum_{(i,j) \in \mathcal{R}(i)^+} (r_{ij} - \mu - b_i - b_j - U_i V_j^T)^2 + \lambda(\|U_i\|_2^2 + \|V_j\|_2^2 + b_i^2 + b_j^2)$
SVD++	$\hat{r}_{ij} = b_{ij} + \Big(U_i + \|N(i)^+\|^{-\frac{1}{2}} \sum_{k \in N(i)^+} y_k\Big) V_j^T$	$\underset{U,V,b_*,y_*}{\arg\min} \sum_{(i,j) \in \mathcal{R}(i)^+} \Big(r_{ij} - b_{ij} - \Big(U_i + \|N(i)^+\|^{-\frac{1}{2}} \sum_{k \in N(i)^+} y_k\Big) V_j^T\Big)^2 + \lambda(\|U_i\|_2^2 + \|V_j\|_2^2 + b_i^2 + b_j^2 + \sum_{k \in N(i)^+} y_k)$
SRui	see [152]	$\underset{U,V}{\arg\min} \frac{1}{2} \sum_{(i,j) \in \mathcal{R}(i)^+} (r_{ij} - U_i V_j^T)^2 + \frac{\alpha}{2} \sum_{i=1}^{m} \sum_{f \in \mathcal{F}^+(i)} s_{if} \|U_i - U_f\|_F^2 + \frac{\beta}{2} \sum_{j=1}^{n} \sum_{q \in \mathcal{Q}^+(j)} s_{jq} \|V_j - V_q\|_F^2 + \frac{\lambda_1}{2} \|U\|_F^2 + \frac{\lambda_2}{2} \|V\|_F^2$
NCRPD-MF	$\hat{r}_{ij} = \mu + b_i + b_j + z + U_i \Big(\frac{1}{\|W_j\|} \sum_{w \in W_j} v_w + \frac{\alpha_1}{\|N_j\|} \sum_{n \in N_j} v_n + \frac{\alpha_2}{\|C_j\|} \sum_{c \in C_j} v_c\Big)^T$	$\underset{U,V,b_*,v_*,\beta_*}{\arg\min} \sum_{(i,j) \in \mathcal{R}(i)^+} (r_{ij} - \hat{r}_{ij})^2 + \lambda_1 \Big(\|U_i\|^2 + \sum_{w \in W_j} \|v_w\|^2\Big) + \lambda_2(b_i^2 + b_j^2 + \beta_i^2 + \beta_j^2) + \lambda_3 \Big(\sum_{n \in N_j} \|v_n\|^2 + \sum_{c \in C_j} \|v_c\|^2\Big)$
FM	$\hat{r}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle x_i x_j$	see [153]

and those of its geographical neighbors (regardless of their business type), revealing that the market environment might play an influential role in an individual business' popularity. After quantifying this correlation with terms, Hu et al. proposed NCRPD-MF, which can incorporate the discovered auxiliary information into BiasSVD (Tab. 5 gives details).

In addition, the strong extensibility of FunkSVD and its variants can also enable them to be integrated with k-nearest neighborhood-based (KNN) recommendation models. For instance, Koren et al. [155] proposed a 3-tier SVD++ model, which can integrate the item-item proximity calculated by KNN models with SVD++. From another perspective, instead of calculating the item-item proximity matrix by similarity metrics (like Pearson correlation coefficient) as KNN models do, Slim [156] learns this matrix by means of optimization algorithms under the framework of FunkSVD (Tab. 6 gives details). Moreover, by applying matrix factorization to learn two embedding matrices P and Q preserving the patterns between items, FISM [157] can estimate the item-item proximity matrix S with P and Q, which is used to be integrated with KNN models. All in all, these integrated methods help out of the dependence on users' co-interactions with items in terms of calculating the item-item proximity matrix S, which has been a fundamental limitation on the accuracy of KNN models as a result of the sparsity problem of recommendation.

However, like any model, FunkSVD and its variants have their critics. Yet much of the criticism is based on the following two flaws. The first one is that the embeddings of users and items learned by the FunkSVD framework could involve negative values, which have difficulties in being well interpreted in practice due to the

general meaninglessness of negative values in reality. One way out of this dilemma is to develop methods of non-negative matrix factorization [158–164]. The other flaw is that the implementation of the FunkSVD framework generally violates the triangle inequality principle [165, 166] because it is put in Hilbert space to measure the user-item proximity by dot-product, which could hinder the preservation of find-grained user preference. To tackle this issue, methods based on metric learning [167, 168] of measuring the user-item proximity can satisfy the triangle inequality principle since they are put in a metric (or Banach) space. In detail, these methods run by constructing a transformed user-item rating matrix R (like by converting a method to convert R into a distance matrix [169]) in the metric space and factorizing it as the FunkSVD framework does [169–173]. Note that the metric space is unnecessary to be Euclidean. For instance, constructing and factorizing the matrix R in hyperbolic space can also work well [174].

In mathematics, most of the aforementioned recommendation models are built on a global low-rank assumption of matrix factorization. Differently, Lee et al. [175] built an assumption that the user-item matrix R is partially observed, which is characterized by a low-rank matrix restricted in the vicinity of certain row-column combinations. Aharon et al. [176] overturned the conventional assumption that a transform matrix should always be observed and fixed. Halko et al. [177] built a randomization assumption, which contributes to a fast matrix factorization on large-scale data. Establishing novel factorization frameworks based on other assumptions from a perspective of mathematics appears to be a challenging, intriguing and promising direction of research into matrix factorization-based recommendation models.

Table 6: Examples of integrating matrix factorization-based models with neighborhood-based collaborative filtering methods. (1) In FISM, $\hat{r}_{ij} = b_i + b_j + (n_i^+ - 1)^{-\alpha} \sum_{k \in \mathcal{R}(i)^+ \setminus \{j\}} P_k Q_j^T$, where n_i^+ is used to control the agreement between the items rated by user i with respect to their respective similarity to item k. (2) In SVD with prior, E is set up to measure the squared loss, absolute loss or generalized Kullback-Liebler divergence. $R(U, V)$ is a regularization term and α is a coefficient used to balance the effects of unobserved ratings.

Model	Objective function
Slim	$\arg\min_{S} \frac{1}{2}\|R - RS\|_F^2 + \frac{\beta}{2}\|S\|_F^2 + \lambda\|S\|_1$
FISMrmse	$\arg\min_{P,Q} \frac{1}{2}\sum_{i,j}\|r_{ij} - \hat{r}_{ij}\|_F^2 + \frac{\beta}{2}(\|P\|_F^2 + \|Q\|_F^2) + \frac{\lambda}{2}\|b_i\|_2^2 + \frac{\gamma}{2}\|b_j\|_2^2$
FISMauc	$\arg\min_{P,Q} \frac{1}{2}\sum_{i} \sum_{j_1 \in \mathcal{R}(i)^+, j_2 \in \mathcal{R}(i)^-} \|(r_{ij_1} - r_{ij_2}) - (\hat{r}_{ij_1} - \hat{r}_{ij_2})\|_F^2 + \frac{\beta}{2}(\|P\|_F^2 + \|Q\|_F^2) + \frac{\gamma}{2}\|b_{j_1}\|_2^2$
SVD with prior	$\underset{U,V}{\text{argmin}} \sum_{(i,j) \in \mathcal{R}(i)^+} E(r_{ij}, U_i V_j^T) + \alpha \sum_{(i,j) \in \mathcal{R}(i)^-} E(\hat{r}_0, U_i V_j^T) + R(U, V)$

3.1.2. Models based on Bayesian analysis

In practice, since the giant amount of users and items while generally very sparse interactions between them in recommender systems, the matrix factorization method could face the non-convex optimization problem [178] when factorizing such a huge and sparse user-item rating matrix, in which case, at best, its recommendation accuracy could largely fluctuate flowing from different model hyper-parameters settings and at worst, its convergence in training (or learning) by optimization algorithms could even be damaged as a result of setting inappropriate model hyper-parameters. As a tool of automatic hyper-parameter adjustment [179, 180], Bayesian analysis method, to some extent, can be used to guide the proper settings of hyper-parameters in matrix factorization-based recommendation models, like by defining regularization terms involved with prior knowledge.

The rationale behind Bayesian analysis-based recommendation can be illuminated by probabilistic matrix factorization (PMF) [181], a probabilistic version of FunkSVD (Tab. 7 gives details). In detail, by hypothesizing that the error $r_{ij} - \hat{r}_{ij}$ obeys the Gaussian distribution as $\mathcal{N}(r_{ij} - U_i V_j^T | 0, \sigma^2)$, where user embedding U_i and item embedding V_i obey the zero-mean spherical Gaussian priors [182], respectively, PMF can maximize the log of the

posterior distribution by

$$\arg\min_{U,V} \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}(r_{ij} - U_i V_j^T)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{M}\|U_i\|_2^2 + \frac{\lambda_V}{2}\sum_{j=1}^{N}\|V_j\|_2^2, \tag{3.1}$$

where $\lambda_U = \frac{\sigma^2}{\sigma_U^2}, \lambda_V = \frac{\sigma^2}{\sigma_V^2}$. Similar in form of the objective function to FunkSVD, Eq. (3.1) is further involved with prior knowledge (i.e., the value ranges of λ_U and λ_V), which can definitely contribute to the settings of hyper-parameters in FunkSVD. However, when inappropriately setting the σ and σ_*^2 in Eq. (3.1), which are still hyper-parameters, PMF could be over-fit in training samples. Faced with this situation, instead of following the hypothesis of PMF that U and V are independent, Bayesian PMF (BPMF) [183] argues that the distributions of U and V are supposed to be non-Gaussian and that λ_U and λ_V can both obey the Gaussian distribution (Tab. 7 gives details). Moreover, in light of the Markov random field [184], Mrf-MF [185] hypothesizes that the prior distributions of U and V should be relevant to user's neighborhood (Tab. 7 gives details). The contributions of Bayesian analysis method are not only throughout the FunkSVD framework but far outside it as well, like those in recommendation based on ordinal data [186] by means of Poisson factorization [187], Bernoulli-Poisson factorization [188] or OrdNMF [163]. Since a widespread tool for hyper-parameter adjustment, Bayesian analysis method can exert its great value in automatic machine learning [179, 180, 189], which has been the focus of recent research into recommendation and other machine learning-based fields.

In addition, Bayesian analysis method can be used to design new strategies for ranking items. Until recently, most recommendation models adopt a point-wise strategy for comparing user's preferences for different items by ranking one's ratings on items, according to their relative size. In recent research, BPR-OPT [190] (based on Bayesian analysis method) pioneered a pair-wise ranking strategy, comparing user's preferences for each pair of different items, in a more fine-grained way. In other words, it hypothesizes that one generally prefers his interacted items more than non-interacted ones (Tab. 7 gives details). Methodologically, for each user i, BPR-OPT builds a training sample $D_S := \{(i, j_m, j_n)|j_m \in \mathcal{R}(i)^+ \wedge j_n \in \mathcal{R}(i)^-\}$, abbreviated as $>_i$, where the tuple (i, j_m, j_n) represents that the user prefers item j_m more than item j_n. Based on the built training samples for all users, maximizing the log of the posterior distribution runs by

$$\arg\min_{\Theta} \sum_{(i,j_m,j_n)\in D_S} -\ln\frac{1}{1+e^{-\hat{x}_{ij_mj_n}}} + \lambda_\Theta\|\Theta\|^2, \tag{3.2}$$

where λ_Θ represents regularization parameters. Besides, the pair-wise ranking strategy can also be integrated with the matrix factorization-based method by, for instance, defining $\hat{x}_{ij} = U_i V_j$ to represent $\hat{x}_{ij_mj_n}$ with $\hat{x}_{ij_m} - \hat{x}_{ij_n}$ [190] under the BPR-OPT framework. So this way, the matrix factorization method, usually oriented to recommendation based on explicit user-item interactions, can be implemented on that based on implicit ones.

As a promising research direction, by means of the prior knowledge of Bayesian analysis method, causal inference [52, 55] aims to understand user's behaviors in recommendation, which contributes to the explainability of recommendation results or even models.

3.1.3. Models based on deep learning

Despite general acknowledgement of the feasibility and effectiveness of matrix factorization and Bayesian analysis methods, they are generally based on shallow learning, only being able to capture and preserve the linear patterns involved in user-item interactions in recommendation. From a mathematical point of view, by representing the features of user i and item j with vectors $v_i \in \mathbb{R}^n$ and $v_j \in \mathbb{R}^n$, respectively, the implementation of recommendation can be described as the process of learning a mapping $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ such that $f(v_i, v_j) = \hat{r_{ij}} \approx r_{ij}$. However, if leaned by matrix factorization method or Bayesian analysis method, the mapping f has to be linear, which is insufficient in fitting any non-linear relation between (v_i, v_j) and r_{ij} (in fact, these relations in practical recommender systems are generally non-linear). Given these inadequacies, recent years have witnessed a boom in applying deep learning methods [192] into recommendation, in order to build deep learning-based recommendation models with non-linear mappings.

Among these models, Youtube Net [193] is a pioneer whose schematics are shown in Fig. 5. Through pre-training, the embeddings of all Youtube videos are learned, based on which the feature vector of each user can be constructed, according to one's video watches. Besides, a user's feature vector is also involved with one's side information like search tokens, geographical information of gender information. After that, the feature vector will be taken as the

Table 7: Examples of modeling Bayesian analysis-based recommendation. (1) In PMF, I_{ij} is an indicator function, equaling to 1 if user i rated item j and 0 otherwise. (2) In BPMF, $\mathcal{W}$ is the Wishart distribution related to the freedom degree ν_0 and a $D \times D$ identity matrix W_0, where $\nu_0 = D$ and $\mu_0 = 0$. (3) In mrf-MF, U_{-i} is a user set where user i is removed. $\tilde{U}_i = \frac{\sum_{i'} K_k^U(i,i') U_{i'}}{K_U}$ where K_U is the number of neighbors, $K_k^U(i,i')$ contains user i's k-nearest neighbors, and $\Sigma_U = I \times \sigma_U^2$. The same is true of items. (4) In BPR-OPT, $\hat{x}_{ij_m j_n}$ is a function of vector Θ, used to represent the relationship among i, j_m and j_n. (5) In RBM, $\tilde{R}_i \in \mathbb{R}^{m \times L}$ is an observed binary indicator matrix of user i, with values $\tilde{R}_{ij}^r = 1$ if the user rated movie j by score r and 0 otherwise.

Model	Prior distribution	Posterior distribution (hypothesis space)	Objective function
PMF	$P(R\|U,V,\sigma^2) = \prod_{i=1}^{M}\prod_{j=1}^{N}[\mathcal{N}(r_{ij}\|U_i V_j^T, \sigma^2)]^{I_{ij}}$, $P(U\|\sigma_U^2) = \prod_{i=1}^{M}\mathcal{N}(U_i\|0,\sigma_U^2 \boldsymbol{I})$, $P(V\|\sigma_V^2) = \prod_{j=1}^{N}\mathcal{N}(V_j\|0,\sigma_V^2 \boldsymbol{I})$	$P(U,V\|R,\sigma^2,\sigma_U^2,\sigma_V^2) \propto P(R\|U,V,\sigma^2)P(U\|\sigma_U^2)P(V\|\sigma_V^2)$	see Eq. (3.1)
BPMF	BPMF gives the prior distributions of μ and σ for PMF: $P(\mu_U,\sigma_U^2 I\|\mu_0,\sigma_0^2 I) = \mathcal{N}(\mu_U\|\mu_0,(\beta_0\sigma_U^2 I)^{-1})\mathcal{W}(\sigma_U^2 I\|W_0,\nu_0)$ $P(\mu_V,\sigma_V^2 I\|\mu_0,\sigma_0^2 I) = \mathcal{N}(\mu_V\|\mu_0,(\beta_0\sigma_V^2 I)^{-1})\mathcal{W}(\sigma_V^2 I\|W_0,\nu_0)$	see [183]	see [183], used Markov Chain Monte Carlo approximation
mrf-MF	$P(U_i\|U_{-i}) = \frac{1}{\sqrt{(2\pi)^{d_U}\Sigma_U}}\exp(-\frac{1}{2}(U_i-\tilde{U}_i)^T\Sigma_U^{-1}(U_i-\tilde{U}_i))$ $P(V_j\|V_{-j}) = \frac{1}{\sqrt{(2\pi)^{d_V}\Sigma_V}}\exp(-\frac{1}{2}(V_j-\tilde{V}_j)^T\Sigma_V^{-1}(V_j-\tilde{V}_j))$	$P(U,V\|\Omega,\Theta) = \frac{P(R^+,U,V\|\Omega,\Theta)}{\iint P(R^+,U,V\|\Omega,\Theta)d_U d_V}$	see [185]
BPR-OPT	$P(>_i\|\Theta) = \prod_{(i,j_m,j_n)\in D_S} P(j_m >_i j_n\|\Theta) = \prod_{(i,j_m,j_n)\in D_S}\frac{1}{1+e^{\hat{x}_{ij_m j_n}(\Theta)}}$, $P(\Theta)\sim N(0,\Sigma_\Theta)$	$P(\Theta\|>_i) \propto P(>_i\|\Theta)P(\Theta)$	see Eq. (3.2)
RBM	$P(\tilde{R}_{ij}^r = 1\|U_i) = \frac{\exp(b_j^r + \sum_{k=1}^{K} U_{ik}W_{jk}^r)}{\sum_{l=1}^{L}\exp(b_j^l + \sum_{k=1}^{K} U_{ik}W_{jk}^l)}$, $P(U_{ik}=1\|\tilde{R}_i) = \sigma(b_k + \sum_{j=1}^{m}\sum_{l=1}^{L} R_{ij}^l W_{jk}^l)$	$P(\tilde{R}_i) = \sum_{U_i}\frac{\exp(-E(\tilde{R}_i,U_i))}{\sum_{\tilde{R}_i',U_i'}\exp(-E(\tilde{R}_i',U_i'))}$	see [191]

input of a deep neural network with multiple layers, used to learn user's embedding. Finally, unobserved implicit user-item interactions can be predicted by using all of the user's embeddings. Researchers have come to see the merits of Youtube Net, including its fast parallel computing [194] and non-linear mapping learning (since it adopts a deep learning framework). Later, its use soon widened to various practical applications, which benefit from its high compatibility of diverse data forms as input.

Under the Youtube Net framework, specific implementations have been proposed. For instance, in order to prepare the embeddings for constructing user's feature vector, neural collaborative filtering (NCF) [195] bases the pre-training on user's implicit interactions with items combined with user's characteristics. The same is true of constructing item's feature vector. After that, as shown in Fig. 5, when coming to predicting an implicit user-item interaction, NCF concatenates the feature vectors of the corresponding user and item, used as the input of a generalized matrix factorization (GMF) layer and multiple MLP layers, respectively, which are both jointed to a NeuMF layer outputting the final prediction. As for being compatible with side information as input, ConvMF [196] proposes an enhanced PMF [181] based on convolution neural networks (CNNs) [197], which can be used to learn the representations of documents. However, these deep learning models could encounter the over-fitting problem. To alleviate that, Cheng et al. [133] proposed to combine deep learning with wide learning [198] as a strategy.

In addition, as for measuring user-item proximity in recommendation, compared with linear operators (like dot product) pervasively adopted by models based on shallow learning, the deep learning framework can be generalized as a non-linear operator for proximity measurement, which is more robust since it could uncover the complex non-linear relations between user-item pairs. For instance, by means of a deep neural network, NCF can learn the non-linear relations between an implicit user-item interaction and the two embeddings of the corresponding user and item. Its implementation was soon generalized to based on explicit user-item interactions by deep matrix factorization (DMF) [199], which can learn the non-linear relations between an explicit user-item interaction and the corresponding values in a user-item matrix Y constructed from both implicit and explicit user-item interactions, as shown in Fig. 5.

One difficulty, however, was that deep learning-based recommendation models generally lack well explainability, since the bedrock of these models is the fitting and optimization theory, long been concerned as almost a black-

box. Faced with this situation, causal learning (or casual inference) [51, 53, 54, 56, 57] seems to be a potential solution, among which restricted Boltzmann machine (RBM) [191] pioneered the application of causal learning in recommendation. As shown in Fig. 5, for a user RBM takes each element of the user's feature vector as an independent unit. Then, it builds the so-called causal relations from these units to the user's interacted items encoded by one-hot, used to model the causality between the user's features and his behaviors (*i.e.*, represented by his interactions with items). As a result, by learning the weights of relations, RBM could unravel how user's features influence his behaviors correspondingly and give them practical meanings oriented to different scenarios, which are definitely helpful to explain the formation of user-item interactions. On the other hand, factorization machine (FM) [153] can be recognized as another pioneer, resorting to casual learning as a strategy for alleviating the sparsity problem in recommendation. In detail, from a fine-grained perspective, FM builds the causal relations between each pair of elements in user's feature vector, named feature interactions, by integrating support vector machine (SVM) with SVD (Tab. 5 gives details). In that case, as a supplement, these discovered causality hidden in user's feature vector contributes to enriching recommendation information. Although RBM and FM are models based on shallow learning, their rationales soon widened to based on deep learning and motivated a variety of deep causal learning-based recommendation models, including DeepFM [200], xDeepFM [201], deep Boltzmann machine [202, 203] and stuff.

3.1.4. Other models

Until recently, in the popular conception it was usually claimed that the user exposure assumption [204] should be the bedrock of recommendation models, which hypothesized that the reason for the existence of unobserved user-item interactions lies in user's limited view of items in recommender systems. In other words, the non-interacted items for a user were those that haven't ever been exposed to the user, thereby being considered valueless as information for recommendation (because no positive or negative preference of the user had any chance for these items). However, this assumption is not invariably true. In recent research, there are conditions under which as Devoogth et al. [205] argued that non-interacted items might not always be beyond a user's view but could be eschewed by the user just as a result of dislikes (the so-called not missing at random assumption [206, 207]) can contradict it. In that case, the user exposure assumption will neglect user's negative preferences for items, which in fact are valuable to be used to construct negative samples [208] for improving training precision. As Devoogth et al. [205] put it, by defining a term $\alpha \sum_{(i,j)\in\mathcal{R}(i)^-} E(\hat{r}_0, U_i V_j^T)$ ($\hat{r}_0$ is a prior estimation on predicted ratings) to measure the probability that an item could be eschewed by a user, the accuracy of the SVD model can be promoted built on the not missing at random assumption (Tab. 6 gives details). In addition, by using a matrix constructed based on the Bernoulli condition, Liang et al. [204] proposed to represent the probability of an item's exposure to a user, as a supplement to recommendation models.

In practice, there are still two flaws in the implementation of the not missing at random assumption. First, this assumption is not fit to implicit user-item interactions because implicit ones only record user's interactions with items without directly carrying user's preferences, which is the so-called positive-unlabeled problem [85, 209, 210]. Second, the general strategy adopted by recommendation models built on the not missing at random assumption is to represent user's negative preferences by equally allocated weights, which could eliminate the deviations in user's negative preferences for different items in reality. In respect to the two flaws, for instance, Saito et al. [211] used a bias to distinguish user's affection degrees toward different items. In addition, by replacing the second term in the objective function of SVD with prior shown in Tab. 6 with $\sum_{(i,j)\in\mathcal{R}(i)^-} c_j E(\hat{r}_0, U_i V_j^T)$, where c_j is the confidence of item j to be non-interacted by users caused by a true negative preference, He et al. [212] proposed a negative weighting allocation strategy considering the popularity of user's non-interacted items.

By bridging two or more recommender systems together and sharing information for recommendation as supplements to either side, cross-domain recommendation [213–215] seems to be a promising direction, which contributes to alleviating the cold start and sparsity problems. In detail, the cross-domain [213] refers to two types of domains: one is target domain in which the final recommendation is directly implemented, and the other is source domain consisting of other recommender systems that could provide observed user-item interactions as supplements corresponding to the unobserved ones in target domain. By utilizing these supplemental user-item interactions from source domain, hidden user-item relations in target domain can be uncovered as enriched observed ones for recommendation. This is sort of similar to the utilization of side information and knowledge into recommendation but is fundamentally different in rationales. For illustration, suppose a user has zero or very few interactions with items in target domain, leading to the cold start and sparsity problems in recommendation. Meanwhile, in source domain if the user ever interacted with sufficient items that also existed in target domain, his preferences for items can still be analyzed and uncovered in source domain, which can be transferred and utilized in target domain to complete

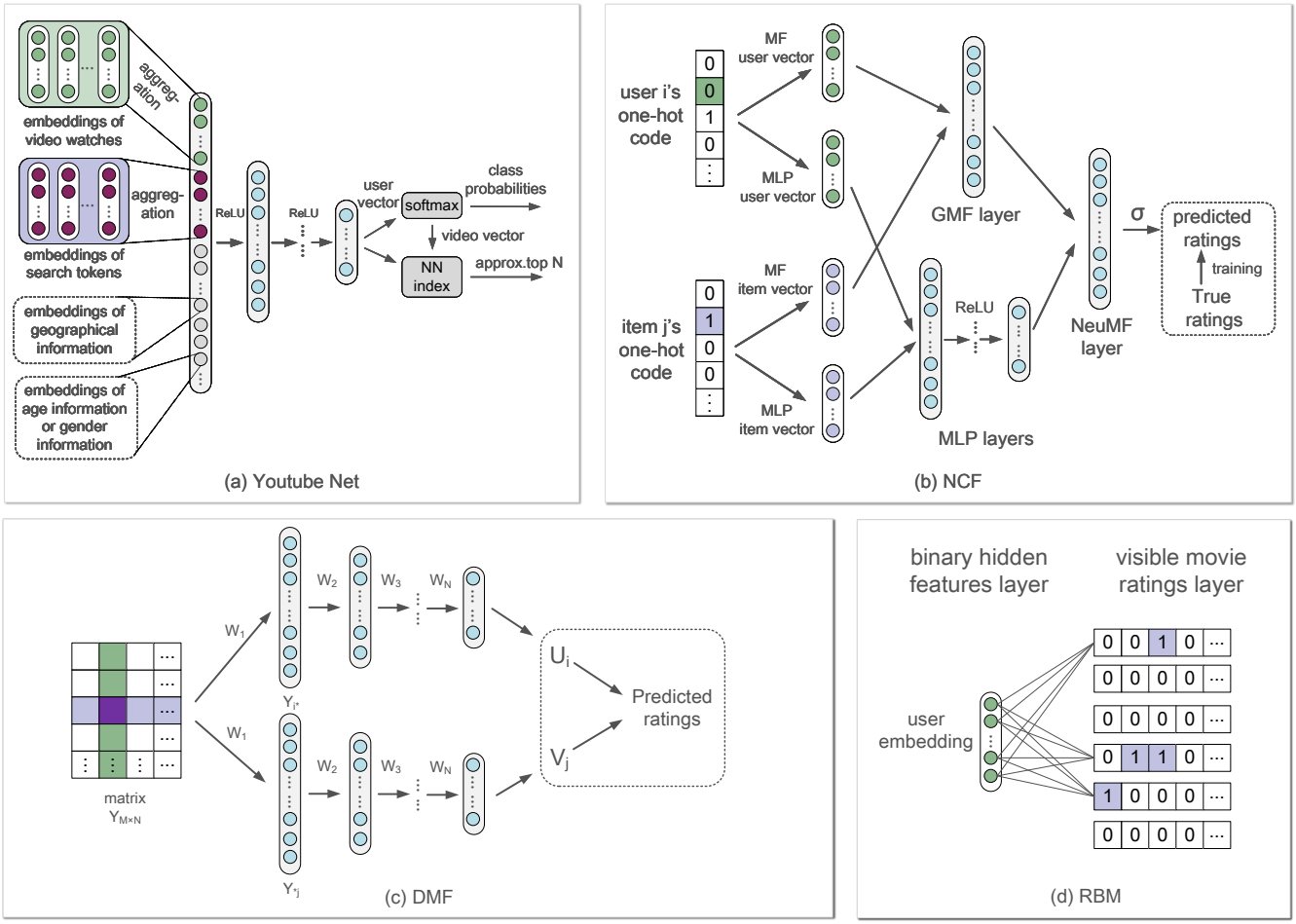

Figure 5: Schematics of Youtube Net, NCF, DMF and RBM. In (a), the generation component for recommendations of Youtube Net is presented, which runs by minimizing the cross-entropy loss with a descent on the output through sampled softmax. In (b), NCF firstly learns four mappings, used to project user i's one-hot code to U_i^{MLP} and U_i^{MF} and to project item j's one-hot code to V_i^{MLP} and V_i^{MF}, respectively. Then, cross-combinations on them are implemented by $\phi^{\text{GMF}} = U_i^{\text{MF}} \odot V_j^{\text{MF}}$ and $\phi^{\text{MLP}} = a_L(W_L^T(a_{L-1}(...a_2(W_2^T(U_i^{\text{MLP}}\ V_j^{\text{MLP}})^T + b_2))...)) + b_L)$, respectively, used as the input of neural networks to learn user i's embedding and item j's embedding, based on which the final prediction can be outputted as $\hat{y}_{ij} = \sigma(h^T(\phi^{\text{GMF}}\ \phi^{\text{MLP}})^T)$. In (c), the corresponding row in matrix Y indexed by user i is mapped to Y_{i*} as user i's embedding U_i, and the same is true of item j's embedding V_j. Based on U_i and V_j, predicting their interaction $S_{ij} = \text{cosine}(U_i, V_j)$ can be performed by minimizing the objective function $L = -\sum_{(i,j)\in\mathcal{R}(i)^+\cup\mathcal{R}(i)^-}(\frac{R_{ij}}{\max(R)}\log\hat{R}_{ij} + (1 - \frac{R_{ij}}{\max(R)})\log(1-\hat{R}_{ij}))$. In (d), RBM has a two-layer architecture based on a neural network framework, where the left layer with K units (*i.e.*, the K elements of user's feature vector) represents user's binary hidden features, and the right layer represents the one-hot encoding of all items. Edges are built between the user and his interacted items, which can be weighted by scores from 1 to L.

recommendation. Methodologically, transfer learning [216–220] is a prevalent technique to realize that mechanism. For instance, in order to transfer user embeddings from source domain to those in target domain, EMCDR [213] learns a one-to-one mapping to bridge the users and items commonly existed in the two domains, in which case the embedding of a cold-start user in target domain can be transferred from that learned in source domain. Besides, based on the assumption that user's preferences for items are almost consistent in different recommender systems, DDTCDR [221] can further support the information exchange between the two domains back and forth by means of dual transfer learning [222–224].

In the past, the theories in optimization algorithms used in recommendation models have escaped widespread investigation. It is only recently when data scale increased dramatically that research into the efficiency of optimization algorithms has attracted any substantial scholar attention, aiming to realize the optimal performance of

recommendation models as quickly as possible while consuming possibly less computing resources. For that purpose, speeding up the learning (*i.e.*, convergence) process of recommendation models plays a large role. Methodologically, for instance, in order to reduce the time complexity of ALS to linearity, He et al. [212] proposed an element-wise ALS (eALS). By switching the constraints and regulation terms in objective functions, Boyd et al. [225] applied ADMM [143] to speed up the optimization process of SLIM. GFNLF [226] achieves a faster convergence process of non-negative matrix factorization (NMF) via adopting $\alpha - \beta-$divergence in objective functions and incorporating a generalized momentum method.

3.2. Recommendation with temporal user-item interactions

Temporal factors of user-item interactions primarily flow from the following situations: (1) New user-item interactions are constantly occurring by existed or new coming user's new interactions with other items unobserved before or newly entered. (2) User's long-term and short-term preferences for items could be changed. As a feasible tool, real-time recommendation [227–229] relies for capturing these temporal factors on dynamically updating the embeddings of users and items, which can learn and preserve user's behavioral changes and contribute to the recommendation accuracy.

The common-used methods of real-time recommendation include the online learning (or online recommendation) method [230, 231], which enables the matrix factorization-based recommendation models to absorb temporal (or newly occurring) user-item interactions in a low computing complexity, and the Markov processes method, which can represent the changes in user's short-term preferences for items.

3.2.1. Models based on matrix factorization

A simple strategy for accommodating newly occurring user-item interactions into recommendation is to reload recommendation models and to repeatedly learn the embeddings of users and items based on the whole interactions combined with the existed and newly occurring ones. Apparently, this strategy is so extremely high in computing resources that could be infeasible in large-scale data. Alternatively, by using only the newly occurring interactions, online learning (or online recommendation) method [230, 231] can directly update the embeddings of users and items based on those previously learned ones. In general, online learning methods implement based on the matrix factorization framework, whose objective functions or optimization frameworks can be perfectly compatible with appended terms representing newly occurring interactions. For instance, as shown in Tab. 8, the objective function of SL with prior can be separated into n blocks, each of which can be used to measure the changes in user's embeddings. Based on the optimization framework of FunkSVD, SGD-PMF and DA-PMF [232] define appended terms to accommodate a newly occurring interaction (u_i, v_j, r_{ij}), incorporated into the updating process of embeddings U_i and V_j (Tab. 8 gives details). So this way, the changes of user's short-term preferences for items can be captured dynamically.

In terms of capturing the changes of user's long-term preferences for items, the matrix factorization framework is also compatible with temporal factors, by setting independent variables of objective functions or optimization algorithms to represent the changes in embeddings of users and items. For instance, as an extension of SVD++, TimeSVD++ [233] extends the terms b_i, b_j and U of SVD++ to $b_i(t)$, $b_j(t)$ and $U_i(t)$ related to time variables. Other methods include moving the time window [234] or setting instance-decay [235].

3.2.2. Models based on Markov processes

Markov processes is another method used to capture the changes of user's short-term preferences, implemented based on analyzing user's sequential activities [91]. Its idea lies in learning an overall-shared transition matrix [238] which can capture and represent the latest user-item interactions (*i.e.*, could reveal user's latest preferences for items), where its elements are transition probabilities between item pairs, that is, the probabilities that a user will interact with each of his unobserved items after having interacted with previously observed ones (*i.e.*, could reveal the probabilities that user's preferences would transition from his interacted items to each of the unobserved ones). Through the learned transition matrix, unobserved interactions can be predicted by ranking the items that mostly meet the transition probability from one's interacted items to unobserved ones. During the whole process, learning an accurate transition matrix is crucial for Markov processes-based recommendation, in which case considering environmental factors [239] in modeling the transition probability seems to contribute a lot.

However, Markov processes-based recommendation models cannot capture the changes of user's long-term preferences. To overcome the flaw, combining the Markov processes framework with the matrix factorization framework is a promising strategy since the latter generally performs well in capturing user's long-term preferences. Methodologically, for instance, by means of the Markov processes framework, FPMC [238] builds several transition matrices

Table 8: Examples of modeling temporal matrix factorization-based recommendation. (1) In SL with prior, $S^V = \sum_j V_j^T V_j$ is a $k \times k$ matrix, which is independent from i. (2) In SGD-PMF, η is the step size controlling the convergence rate during updating iterations. (3) In DA-PMF, Y_{U_i} is the approximation of $(\sum(\hat{r}_{ij} - r_{ij})\hat{r}'_{ij}V_j)/|\mathcal{R}(i)^+|$.

Model	Objective function	Optimization
SL with prior	$\underset{U_i,V_j}{\operatorname{argmin}} \sum_{i\in\mathcal{R}(i)^+} \sum_{j\in\mathcal{R}(i)^+} [(r_{ij} - U_i V_j^T)^2 - \alpha(U_i V_j^T)^2] + \alpha U_i S^V U_i^T$	using randomized block coordinate descant [236] and line search [237]
SGD-PMF	$\underset{U_i,V_j}{\operatorname{argmin}} \ (r_{ij} - \hat{r}_{ij})^2 + \frac{\lambda_U}{2}\|U_i\|_2^2 + \frac{\lambda_V}{2}\|V_j\|_2^2$	$U_i \leftarrow U_i - \eta((\hat{r}_{ij} - r_{ij})\hat{r}'_{ij}V_j + \lambda_U U_i)$ $V_j \leftarrow V_j - \eta((\hat{r}_{ij} - r_{ij})\hat{r}'_{ij}U_i + \lambda_V V_j)$
DA-PMF	$\underset{U_i,V_j}{\operatorname{argmin}} \ (r_{ij} - \hat{r}_{ij})^2 + \frac{\lambda_U}{2}\|U_i\|_2^2 + \frac{\lambda_V}{2}\|V_j\|_2^2$	$Y_{U_i} \leftarrow \frac{\|\mathcal{R}(i)^+\| - 1}{\|\mathcal{R}(i)^+\|} Y_{U_i} + \frac{1}{\|\mathcal{R}(i)^+\|}(\hat{r}_{ij} - r_{ij})\hat{r}'_{ij}V_j$ $Y_{V_j} \leftarrow \frac{\|\mathcal{N}(j)^+\| - 1}{\|\mathcal{N}(j)^+\|} Y_{V_j} + \frac{1}{\|\mathcal{N}(j)^+\|}(\hat{r}_{ij} - r_{ij})\hat{r}'_{ij}U_i$ $U_i = \underset{\omega}{\operatorname{argmin}}\{Y_{U_i}^T\omega + \lambda_U\|\omega\|_2^2\}$ $V_j = \underset{\omega}{\operatorname{argmin}}\{Y_{V_j}^T\omega + \lambda_V\|\omega\|_2^2\}$

Table 9: Examples of modeling Markov processes-based recommendation. (1) In Fossil, the model can be reduced to the first-order when $L = 1$. (2) In MFMP, $X_i(t) \sim \mathcal{N}(X_i(t)|0, \sigma_U^2\boldsymbol{I})$ and $Y_j(t) \sim \mathcal{N}(Y_j(t)|0, \sigma_V^2\boldsymbol{I})$. The parameters in Eq. (3.3) are defined as $\rho_U := \sigma^2/\Sigma_U^2$, $\rho_V := \sigma^2/\Sigma_V^2$, $\lambda_U := \lambda^2/\sigma_U^2$ and $\lambda_V := \sigma^2/\sigma_V^2$.

Model	Prior distribution	Posterior distribution (hypothesis space)	Loss function
Fossil	see [240]	$P_i(j\|\mathcal{R}(i)^+_{t-1}, \mathcal{R}(i)^+_{t-2}, \ldots, \mathcal{R}(i)^+_{t-L}) \propto \beta_j + \langle \frac{1}{\|\mathcal{R}(i)^+ \setminus \{j\}\|^\alpha} \sum_{k\in\mathcal{R}(i)^+\setminus\{j\}} P_k + \sum_{l=1}^{L}(\eta_l + \eta_l^i)\cdot P_{\mathcal{R}(i)^+_{t-l}}, Q_j\rangle$	S-BPR [240]
MFMP	$P(R(t)\|U(t), V(t)) = \prod_{t=0}^{T}\prod_{i,j\in\mathcal{R}(i)^+_t} \mathcal{N}(r_{ij}(t)\|U_i(t)V_j(t)^T, \sigma^2)$ $U_i(t+1) = U_i(t) + X_i(t), \quad V_j(t+1) = V_j(t) + Y_j(t)$ $P(U_i(0)) = \mathcal{N}(U_i(0)\|0, \Sigma_U^2\boldsymbol{I}), \quad P(V_j(0)) = \mathcal{N}(V_j(0)\|0, \Sigma_V^2\boldsymbol{I})$	$P(U(t), V(t)\|R(t))$	see Eq. (3.3)

corresponding to each user used to aggregate into a tensor, where the missing values correspond to unobserved user-item interactions. After that, through Tucker decomposition (TD) [241], FPMC factorizes the tensor into the embedding matrices of users and items, used to approximately reconstruct the original tensor in order to predict the missing values, as the matrix factorization framework does. Besides, by fusing user's long-term preferences learned by the matrix factorization framework and user's short-term preferences captured by a high-order Markov chain, Fossil [240] can represent user's hybrid preferences for items. By applying embedding techniques to build the transition matrices of the Markov processes framework, Wu et al. [242] enabled the transition probabilities to involve user's long-term preferences. Since most of these works fuse user's long-term and short-term preferences linearly, they inevitably lose the higher-order patterns hidden in user-item interactions. In this regard, Wang et al. [243] proposed a two-layer structure constructed with different aggregation operations.

In addition, based automatic hyper-parameters adjustment on Bayesian analysis method, the accuracy of Markov processes-based recommendation models can be enhanced to some extent. Take MFMP [244] as an instance, the probabilistic version of TimeSVD++ [233]. By hypothesizing that the changes of $U_i(t)$ and $V_j(t)$ over time follow the Gaussian Hidden Markov processes rule, maximizing the posterior distribution can run by

$$\begin{aligned}\arg\min_{U(t),V(t)} &\sum_{t=0}^{T}\sum_{(i,j)\in\mathcal{K}(t)}\left(R_{ij}(t) - U_i(t)V_j(t)^T\right)^2 + \rho_U\|U(0)\|_2^2 + \rho_V\|V(0)\|_2^2 \\ &+ \lambda_U\sum_{t=1}^{T}\|U(t) - U(t-1)\|_2^2 + \lambda_V\sum_{t=1}^{T}\|V(t) - V(t-1)\|_2^2.\end{aligned} \tag{3.3}$$

3.3. Summary

The key developments of bipartite graph embedding-based recommendation models retrospected in this section are concisely summarized in Fig. 6, among which it is overt that the matrix factorization framework has long been concerned as the most extensible one succeeding various subsequent variants. However, the current focus has substituted the deep learning framework for the matrix factorization framework, for the advantages in non-linear pattern discovery and parallel computing of the former one is meeting the technical requirements of increasing complexity and scale of data for recommendation. As for the Bayesian analysis framework, its breakthrough runs through the entire timeline, indicating its indispensable contributes to the other methods for automatic hyper-parameter adjustment. On the other hand, the Markov method does not seem to be common in use.

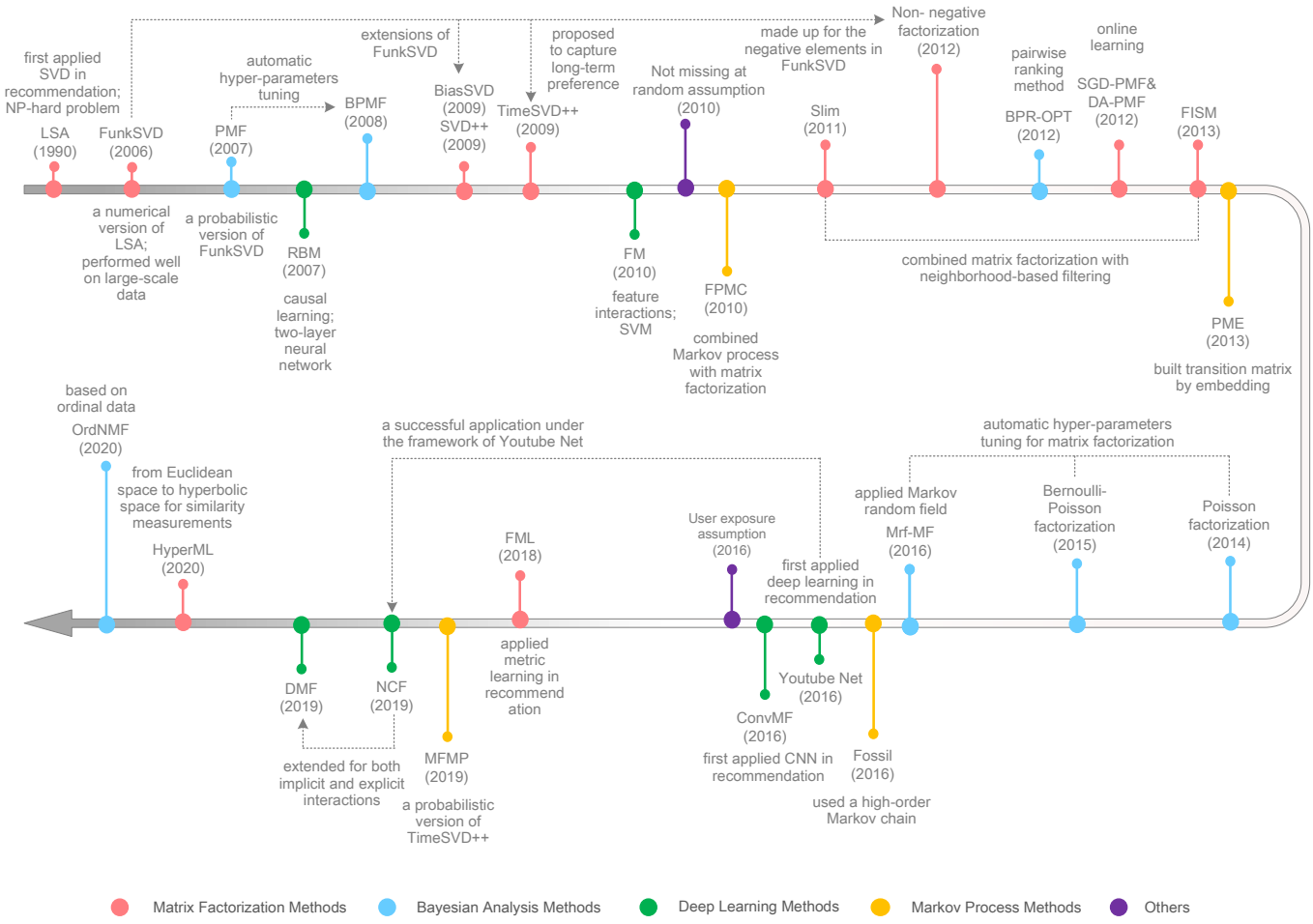

Figure 6: Timeline of key developments in bipartite graph embedding recommendation.

4. General graph embedding for recommendation

As illustrated in Sec. 2, employing side information (like the properties of users and items) in uncovering hidden (indirect) user-item relations can alleviate the cold start and sparsity problems, which contributes to the recommendation accuracy. However, since side information is usually represented by general graphs which are far beyond bipartite graphs in complexity, the matrix factorization method as well as other methods designed for bipartite graph embedding face fundamental limits on the practice of the recommendation involving side information. To make up for the flaw, until recently, researchers have tried to extend the matrix factorization method for general

graph embedding, like methods of collective matrix factorization [245, 246] and spectral [247–250]. Although being feasible, these new methods still cannot run efficiently enough on large-scale data attributed to their high computing complexity.

In recent research, in order to uplift the model scalability on general graphs with large-scale, general graph embedding techniques have been developed. In the first place, Sec. 4.1 divides these techniques into three categories by those based on methods of translation, meta path and deep learning. Based on that, Sec. 4.2 retrospects their corresponding application in recommendation from two different perspectives: technique-oriented and scenario-oriented.

4.1. Three categories of techniques for general graph embedding

Techniques of general graph embedding can be divided into three categories: those based on methods of translation, meta path and deep learning. From an overview, built on algebraic theory, the translation-based techniques have come to see the merit of being able to sufficiently preserve local topological features in a graph. To further capture a graph's global topological features, the Meta path-based techniques are run by random walking across nodes. Being similar in mechanism to the deep learning-based models for bipartite graph embedding, the deep learning techniques can also capture and preserve the non-linear topological features hidden in general graphs.

4.1.1. Techniques based on translation

The triplets (h, r, t) illustrated in Sec. 2.1.2 describes a relation r from the head node (or entity) h to the tail node t, whose embeddings are denoted by $\boldsymbol{h}, \boldsymbol{r}$ and $\boldsymbol{t}$, respectively.

From the perspective of algebraic theory, TransE [251] takes $\boldsymbol{r}$ as a translation from $\boldsymbol{h}$ to $\boldsymbol{t}$ in a metric space, which satisfies that when (h, r, t) existed $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$ indicating that $\boldsymbol{t}$ is supposed to be one of the nearest neighbors of $\boldsymbol{h} + \boldsymbol{r}$ while $\boldsymbol{h} + \boldsymbol{r}$ should be far away from $\boldsymbol{t}$ otherwise. However, in practice, a node shared by multiple communities in a general graph could play different roles like, for example, that a head node h could have positive impacts on other tail nodes in a community (*e.g.*, a boss could bring interests to the staffs of his own incorporation) but may have negative impacts on another community (*e.g.*, the boss could earn some profits from the staffs of competitive incorporation), in which case h should be denoted by h_0 and h_1 oriented to its different roles (*i.e.*, impacts), respectively. Consequently, representing h_0 and h_1 by a common embedding $\boldsymbol{h}$ as TransE does would not be able to distinguish the different practical meanings. TransH [252] gives clues to out of this dilemma. Based on the assumption that a node's distinctive roles of different communities in a general graph could be revealed (or represented) by its diverse relations with other nodes (*i.e.*, the positive relations of the node with other nodes could reveal its positive role in the related community and vice versa), TransH maps each pair of (h, t) to multiple relation-specific hyper-planes $\boldsymbol{w_r}$, which are used to represent the diverse relations between nodes. Analogously, in a general graph (especially a heterogeneous graph), a relation between two nodes could also play multiple roles. For example, a triplet (location, contains, location) can be interpreted by multiple semantics such as country-contains-city, country-contains-university or something. Consequently, representing the relation by one embedding $\boldsymbol{r}$ is insufficient in distinguishing such abundant relational semantics. In this regard, TransR [67] separately maps nodes and relations into a node space and different relation spaces corresponding to the diverse relational semantics between head-tail pairs, respectively. Moreover, on the advice of the diverse semantics of a specific relation r, CTransR [67] clusters r's linked node pairs (h, t) into multiple groups, corresponding to the different semantics of r. In these ways, TransR and CTransR can preserve multiplex semantics of nodes and relations in a general graph. Tab. 10 gives details of these techniques.

Besides, in a general graph, the multiplicity (*i.e.*, diverse types or appended attributes) of nodes and relations is far beyond their multiple semantics, which plays a large role in representing a general graph as precisely as possible. In view of that, TransD [253] takes the multiple types of nodes and relations into account. Ji et al. [68] proposed that relations could belong to different graph patterns like that some of them are linked with a large number of node pairs while others may not. In addition, Ji et al. also discovered that relations could have uneven balances that the quantity of node pairs linked with relations could differ a lot. Moreover, by means of the Chinese restaurant process (CRP) [254], TransG [255] can cluster the semantic components $\pi_{r,m}$ in (h, r, t) of nodes and relations. Tab. 10 gives details of these techniques.

An efficient objective function for the above translation-based techniques is

$$L = \sum_{(h,r,t)\in S} \sum_{(h',r',t')\in S'} [f_r(\boldsymbol{h}, \boldsymbol{t}) + \gamma - f_r(\boldsymbol{h'}, \boldsymbol{t'})]_+, \tag{4.1}$$

which can be run by minimizing a margin-based ranking criterion in optimization, where $[x]_+$ denotes the positive part of x, and $\gamma > 0$ is a margin hyper-parameter. $f_r(\boldsymbol{h}, \boldsymbol{t})$ is the loss function such as those shown in Tab. 10. $S = \{(h, r, t)|h, t \in E\}$ is the golden triplets set or named training samples, and $S' = \{(h', r, t)|h' \in E\} \cup \{(h, r, t')|t' \in E\}$ is the negative triplets set or named negative samples, whose efficiency could largely influence the training process, which is generally constructed by replacing either the head node or the tail node with a negative one selected by random (but usually not both at the same time).

Nevertheless, since built on the mechanism of algebraic transformation on local topology, the above translation-based techniques could lose the global topological features in a general graph. In this case, researchers resorted to deepening the proximity order of transformation as a strategy for capturing and preserving global topology under the translation-based framework. For instance, by building a path space used to preserve relational semantics, RPE [256] can measure the higher-order proximity of non-adjacent node pairs connected by paths in the space. However, a limitation of extending the translation-based framework to higher-order ones is clear: high computing complexity. In order to reach higher efficiency in representing the global topological features in a general graph, meta path-based techniques are hitherto prevalent, as illustrated in the next section.

Table 10: Examples of modeling translation-based techniques (1) In TransH, after being mapped to a hyper-plane $\boldsymbol{w_r}$, the embeddings of nodes h and t in a pair (h, t) are denoted by $\boldsymbol{h}_\perp$ and $\boldsymbol{t}_\perp$, respectively. Correspondingly, the embedding of the relation between them on $\boldsymbol{w_r}$ with a specific meaning is denoted by $\boldsymbol{d_r}$. (2) In TransR, $\boldsymbol{M_r}$ is a transformation matrix used to map a head-tail pair with a relation r into different relation-specific spaces corresponding to distinctive relationsl semantics. (3) In TransD, $\boldsymbol{h_p}, \boldsymbol{t_p} \in \mathbb{R}^n$ represent the semantics of head node and tail node, respectively, and $\boldsymbol{r_p} \in \mathbb{R}^m$ represents the semantics of the relation between them. $\boldsymbol{M_{rh}}, \boldsymbol{M_{rt}} \in \mathbb{R}^{m \times n}$ are two transformation matrices. (4) In TranSparse, θ measures the sparse degree of a matrix, recording the fraction of zero elements over the total number of elements. For each relation $\boldsymbol{r}$, a sparse transfer matrix $\boldsymbol{M_r}(\theta) \in \mathbb{R}^{m \times n}$ is constructed. $0 \leq \theta_{\min} \leq 1$ is a hyper-parameter, which denotes the minimum sparse degree, N_r denotes the number of node pairs linked by the relation r, and N_{r*} denotes the maximum number in N_r.

Technique	Mapping	Loss function
TransE	none	$f_r(\boldsymbol{h}, \boldsymbol{t}) = \|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|_2^2$
TransH	$\boldsymbol{h}_\perp = \boldsymbol{h} - \boldsymbol{w}_r^T \boldsymbol{h} \boldsymbol{w}_r, \boldsymbol{t}_\perp = \boldsymbol{t} - \boldsymbol{w}_r^T \boldsymbol{t} \boldsymbol{w}_r$	$f_r(\boldsymbol{h}, \boldsymbol{t}) = \|\boldsymbol{h}_\perp + \boldsymbol{r} - \boldsymbol{t}_\perp\|_2^2$
TransR	$\boldsymbol{h}_r = \boldsymbol{h}\boldsymbol{M}_r, \boldsymbol{t}_r = \boldsymbol{t}\boldsymbol{M}_r$	$f_r(\boldsymbol{h}, \boldsymbol{t}) = \|\boldsymbol{h}_r + \boldsymbol{r} - \boldsymbol{t}_r\|_2^2$
CTransR	$\boldsymbol{h}_{r,c} = \boldsymbol{h}\boldsymbol{M}_r, \boldsymbol{t}_{r,c} = \boldsymbol{t}\boldsymbol{M}_r$	$f_r(\boldsymbol{h}, \boldsymbol{t}) = \|\boldsymbol{h}_{r,c} + \boldsymbol{r_c} - \boldsymbol{t}_{r,c}\|_2^2 + \alpha\|\boldsymbol{r}_c - \boldsymbol{r}\|_2^2$
TransD	$\boldsymbol{M}_{rh} = \boldsymbol{r}_p \boldsymbol{h}_p^T + \boldsymbol{I}^{m \times n}, \boldsymbol{M}_{rt} = \boldsymbol{r}_p \boldsymbol{t}_p^T + \boldsymbol{I}^{m \times n}$ $\boldsymbol{h}_\perp = \boldsymbol{M}_{rh}\boldsymbol{h}, \boldsymbol{t}_\perp = \boldsymbol{M}_{rt}\boldsymbol{t}$	$f_r(\boldsymbol{h}, \boldsymbol{t}) = -\|\boldsymbol{h}_\perp + \boldsymbol{r} - \boldsymbol{t}_\perp\|_2^2$
TranSparse(share)	$\theta_r = 1 - (1 - \theta_{\min})N_r/N_{r*}$ $\boldsymbol{h}_p = \boldsymbol{M}_r(\theta_r)\boldsymbol{h}, \boldsymbol{t}_p = \boldsymbol{M}_r(\theta_r)\boldsymbol{t}$	$f_r(\boldsymbol{h}, \boldsymbol{t}) = \|\boldsymbol{h}_p + \boldsymbol{r} - \boldsymbol{t}_p\|_2^2$
TranSparse(separate)	$\theta_r^l = 1 - (1 - \theta_{\min} N_r^l/N_{r*})^{l^*} (l = h, t)$ $\boldsymbol{h}_p = \boldsymbol{M}_r^h(\theta_r^h)\boldsymbol{h}, \boldsymbol{t}_p = \boldsymbol{M}_r^t(\theta_r^t)\boldsymbol{t}$	$f_r(\boldsymbol{h}, \boldsymbol{t}) = \|\boldsymbol{h}_p + \boldsymbol{r} - \boldsymbol{t}_p\|_2^2$
TransG	see [255]	$P\{(h, r, t)\} \propto \sum_{m=1}^{M_r} \pi_{r,m} P(\boldsymbol{u_{r,m}}\|h, t)$ $= \sum_{m=1}^{M_r} \pi_{r,m} e^{-\frac{\|\boldsymbol{u_h} + \boldsymbol{u_{r,m}} - \boldsymbol{u_t}\|_2^2}{\sigma_h^2 + \sigma_t^2}}$

4.1.2. Techniques based on meta path

Meta path-based techniques were inspired from the basic thought of word representation [257] in natural language processing (NLP). Given N words chosen from a corpus containing hundreds of millions of words, a legal sentence like $w_1, w_2, ..., w_N$ (w_* represents a word) with effective meanings can be made up with them. Since words appearing in the same context of a sentence are hypothesized to be similar in meanings, the representations (i.e., embeddings) of these words are supposed to be close in proximity. Methodologically, by taking legal sentences as training samples, NPL [258] can learn the representation of a given word, which could involve the word's context (Fig. 7 gives details). From a reverse perspective, Skip-gram [259] aims to learn the representations of words in a given context around the context's central word, which could capture and preserve the linguistic patterns among these words (Fig. 7 gives details). Compared with NPL, Skip-gram appears to be more available for large-scale data. Furthermore, by using

hierarchical softmax [260] to identify phrases, Word2vec [261] can extend Skip-gram from a word-based model to a phrase-based one. Tab. 11 gives details of these models.

In the same way, if taking a graph as the analogy of a corpus, the thought of word representation can be applied in graph embedding. Given N nodes picked up from a graph by random walking which starts or ends at an arbitrary node, a nodal sequence can be made up with them. After repeating random walking multiple times, a set of nodal sequences can be generated. Much as words in sentences, the nodes appearing in a nodal sequence with higher frequency are hypothesized to be represented by embeddings with closer proximity. The feasibility of this transferred thought was firstly proven by DeepWalk [262, 263]. In detail, Deepwalk adopts the depth-first searching strategy for generating random walks $\mathcal{W}_{v_r} = (\mathcal{W}^1_{v_r}, \mathcal{W}^2_{v_r}, ..., \mathcal{W}^k_{v_r})$ rooted at an arbitrary node v_r, where $\mathcal{W}^{k+1}_{v_r}$ is a node randomly picked up from $\mathcal{W}^k_{v_r}$'s neighbors. In reality, path $\mathcal{W}_{v_r}$ can be recognized as a particular "sentence". After denoting the embedding of each node $v_i \in \mathcal{W}_{v_r}$ by $\boldsymbol{v_i} \in \mathbb{R}^d$ and denoting its surrounding context by $v_j \in \mathcal{W}_{v_r}[i-w : i+w]$ (where the window size is $2\omega+1$), the training process can be run by maximizing $P(v_j|v_i)$ with hierarchical softmax [260] (Tab. 11 gives details). So this way, DeepWalk can well capture and preserve the high-order proximity between nodes. On the other hand, by adopting the breadth-first searching strategy for generating random walks, LINE [264] can concentrate on the first-order and second-order proximity between nodes, which is generally called "WideWalk" (Tab. 11 gives details).

However, being in sharp contrast to word representation whose legitimacy of constructed sentences generally can be guaranteed by human linguistic knowledge, the legitimacy of random walks still lacks a recognized examination standard, which could hurt the accuracy of learned embeddings since inaccurate or even incorrect random walks are insufficient to capture a graph's global topology. Under this condition, based on the frameworks of DeepWalk or LINE, designing more intelligent random walking rules has been the focus of subsequent research into meta path-based techniques. For instance, by defining a more flexible notion of node's neighbors, Node2vec [69] equips random walking with abilities in identifying nodes of a common community or of similar roles in a general graph. In detail, Node2vec designs a biased random walking rule guided by a return parameter p and an in-out parameter q as

$$P(v_i|v_{i-1}) = \frac{1}{Z}\alpha_{pq}(i-2,i)\cdot\omega_{i-1,i}, \quad \alpha_{pq}(i-2,i) = \begin{cases} 1/p & \text{if } d_{i-2,i}=0, \\ 1 & \text{if } d_{i-2,i}=1, \\ 1/q & \text{if } d_{i-2,i}=2, \end{cases} \tag{4.2}$$

where v_{i-1} walks from v_{i-2}, v_i is a neighbor of v_{i-1}, $d_{i-2,i}$ is the shortest path between v_{i-2} and v_i, Z is a normalization constant, and $\omega_{i-1,i}$ is the weight of edge $(i-1,i)$ (Tab. 11 gives details). Moreover, as for designing new random walking rules from a deeper perspective in algebra, NetMF [265] unifies DeepWalk, LINE and Node2vec into a matrix factorization framework.

In practice, as illustrated in Sec. 4.1.1, nodes and relations in general graphs carry diverse semantics, types and attributes. However, the above techniques are basically oriented to homogeneous graphs. In order to capture and preserve the multiplicity of heterogeneous graphs (most general graphs are heterogeneous ones), for instance, PTE [266] extends LINE to be available on bipartite graphs (Tab. 11 gives details). In addition, by decomposing a bipartite graph into two homogeneous graphs, BiNE [267] designs a random walking rule of "rich nodes are getting richer", in order to satisfy the power-law distribution phenomenon. Recently, by means of manually built meta paths [268, 269] based on expert knowledge, designing random walking rules on heterogeneous graphs has been a promising solution, which aims to cover the multiplicity by meta paths as sufficient as possible. HIN2Vec [270] is a case in point. By capturing multiple relational types between nodes, HIN2Vec can jointly learn the embeddings of nodes based on training samples $\langle u, v, r, L(u,v,r)\rangle$, which indicate that a relation r exists between u and v when $\langle u, v, r\rangle = 1$ and vice versa. Moreover, during random walking, Metapath2vec/ Metapath2vec++ [271] can capture the diverse nodal and relational types in meta paths by defining a translation probability between nodes as

$$P(v_i|v^{t-1}_{i-1}) = \begin{cases} \dfrac{1}{|N_t(v^{t-1}_{i-1})|} & (v_i, v^{t-1}_{i-1}) \in E, \phi(v_i) = t, \\ 0 & (v_i, v^{t-1}_{i-1}) \in E, \phi(v_i) \neq t, \\ 0 & (v_i, v^{t-1}_{i-1}) \notin E. \end{cases} \tag{4.3}$$

Where $v^{t-1}_{i-1} \in V_{t-1}$, $N_t(v^{t-1}_{i-1})$ denotes the V_t type of v^{t-1}_{i-1}'s neighborhood and $v_i \in V_t$. Besides, there are also techniques [272–274] used to represent the multiplicity of relations in general graphs.

Table 11: Examples of modeling meta path-based techniques. (1) In Skip-gram, w_O and w_I denote a word w's input and output embedding, respectively. N is the number of words in a corpus, L is the number of words in a sentence (*i.e.*, a training sample) and c is the size of a context. (2) In Word2vec, N' is the number of negative samples, and $P_n(w)$ is a noise distribution. (3) In LINE, the embedding of v is denoted by $\boldsymbol{v}$ when treating v as a vertex while by $\boldsymbol{v}'$ when a "context". (4) In PTE, nodes in a bipartite graph are divided into two sets V_A and V_B. $P(v_j|v_i)$ denotes the conditional probability that node v_j in set V_B is walked from node v_i in set V_A.

Technique	Random walk strategy	Loss function
Skip-gram	depth-first search	$P(w_O\|w_I) = \frac{\exp(\boldsymbol{w_O^T w_I})}{\sum_{k=1}^{N} \exp(\boldsymbol{w_{kO}^T w_I})}$
Word2vec	depth-first search	$P(w_O\|w_I) = \log \sigma(\boldsymbol{w_O^T w_I}) + \sum_{k=1}^{N'} \mathbb{E}_{w_k \sim P_n(w)}[\log \sigma(-\boldsymbol{w_{kO}^T w_I})]$
DeepWalk	depth-first search	$P(u_j\|v_i) = \prod_{l=1}^{\lceil \log \|V\| \rceil} P(b_l\|v_i)$
LINE	breadth-first search	$P_1(v_i, v_j) = \frac{1}{1+\exp(-\boldsymbol{v_i} \cdot \boldsymbol{v_j})}$, $P_2(v_j\|v_i) = \log \sigma(\boldsymbol{v_j'^T} \cdot \boldsymbol{v_i}) + \sum_{k=1}^{N'} \mathbb{E}_{v_k \sim P_n(v)}[\log \sigma(-\boldsymbol{v_k'^T} \cdot \boldsymbol{v_i})]$
PTE	breadth-first search	$P(v_j\|v_i) = \frac{\exp(\boldsymbol{v_j^T} \cdot \boldsymbol{v_i})}{\sum_{k \in V_B} \exp(\boldsymbol{v_k^T} \cdot \boldsymbol{v_i})}$
Node2vec	see Eq. (4.2)	$P(v_j\|v_i) = \frac{\exp(\boldsymbol{v_j^T} \cdot \boldsymbol{v_i})}{\sum_{k \in V} \exp(\boldsymbol{v_k^T} \cdot \boldsymbol{v_i})}$
Metapath2vec	see Eq. (4.3)	$P(v_j\|v_i) = \frac{\exp(\boldsymbol{v_j^T v_i})}{\sum_{k \in V} \exp(\boldsymbol{v_k^T v_i})}$
Metapath2vec++	see Eq. (4.3)	$P(v_{j_t}\|v_i) = \frac{\exp(\boldsymbol{v_{j_t}^T v_i})}{\sum_{k_t \in V_t} \exp(\boldsymbol{v_{k_t}^T v_i})}$

4.1.3. Techniques based on deep learning

As illustrated in Sec. 3.1.3 that deep learning methods are equipped with abilities in capturing the non-linear features in a bipartite graph, by the same token it is also true of those in a general graph. Among them, AutoEncoder [277] is a representative one, an unsupervised deep learning framework, which is different from those based on supervised deep learning frameworks in Sec. 3.1.3. In detail, as shown in Fig. 7, AutoEncoder consists of two components: the Encoder and the Decoder. The Encoder can learn the input X's embedding, which will be stored in the hidden layer Z. Then, taking the learned embedding as input, the Decoder is used to reconstruct X by Y, which is supposed to be approximate to X as much as possible, by minimizing the reconstruction error between X and Y. Subsequently, this framework has matured as different variants. For instance, by corrupting the input X to $\tilde{X}$ and minimizing the reconstruction error between $\tilde{X}$ and Y, Vincent et al. [278] can enhance the robustness of the learned embedding (*i.e.*, Z) (Fig. 7 gives details). Salakhutdinov et al. [279] proposed a multi-layer version of AutoEncoder named Deep AutoEncoder. More variants of AutoEncoder include SCAE [280], generalized AutoEncoder (GAE) [281], variational AutoEncoders (VAEs) [282] and deep hierarchical variational AutoEncoder (Nvae) [283], to name a few. When it comes to generalizing the AutoEncoder framework and its variants to general graph embedding, SDNE [284] provides a semi-supervised deep model, preserving the second-order proximity between node pairs by reconstructing their common neighborhoods with two deep AutoEncoders sharing common parameters. Meanwhile, SDNE can also preserve the first-order proximity between nodes by using a Laplacian Eigenmaps-based supervised component.

However, one fundamental limitation of the AutoEncoder framework is that the dimension of hidden layer Z is fixed, and so are its variants. In truth, if the input X's dimension is far higher than that of Z, from X to Z essential information could be lost through the Encoder, which is insufficient to represent the original X. For instance, as shown in Fig. 8, if the input sequence $(x_1, x_2, ..., x_n)$ of Seq2seq [285] is high in dimension, compressing $z_1, z_2, ..., z_n$ (generated from $(x_1, x_2, ..., x_n)$ correspondingly) into an embedding with fixed dimension could lose essential patterns of correlation among X's elements. To deal with the issue, by means of the attention mechanism [286–289], attention weights a_* for z_* of Seq2seq can be learned, used to quantify z_*'s importance. In other words,

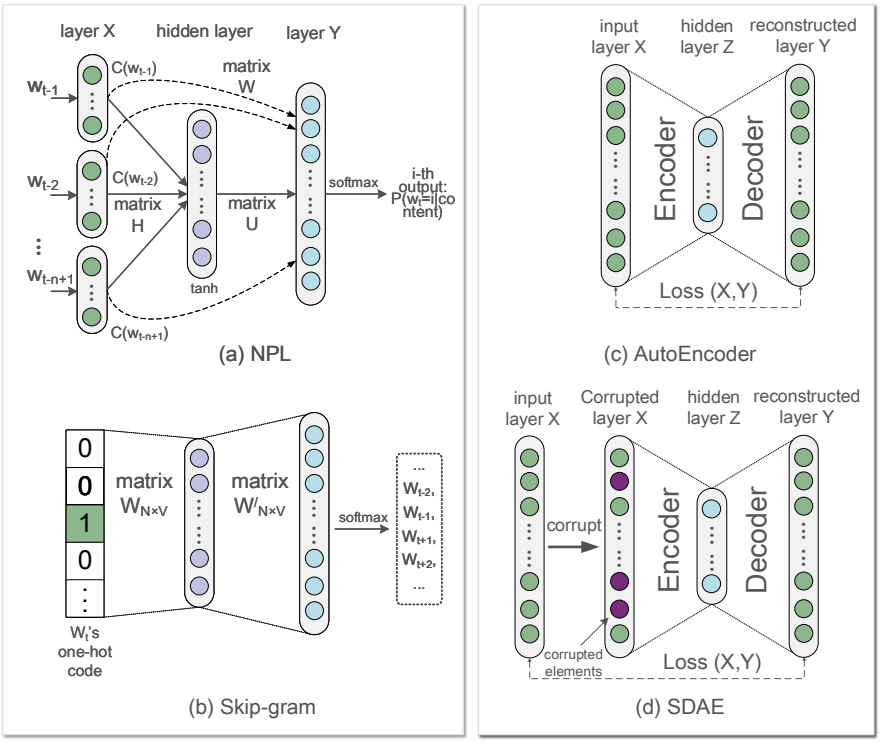

Figure 7: Schematics of NPL, Skip-gram, AutoEncoder and SDAE. In (a), a legal sentence $w_t, w_{t-1}, ..., w_{t-n+1}$ is made up with n sequential words, where w_* records a word's index in a corpus. NPL takes the sentence as a training sample and transforms it into $x = (C(w_{t-1}), C(w_{t-2}), ..., C(w_{t-n+1}))$, where $C(w_*)$ is the one-hot code of w_*. The training process can be run by maximizing $P(w_t = \alpha | w_{t-1}, ..., w_{t-n+1}) = \text{softmax}(b + Wx + U \tanh(d + Hx))$, which represents the probability of w_t's index to be α. In (b), from a reverse perspective, by taking w_t as the input, Skip-gram can reconstruct w_t's surrounding context as the output. In (c), according to different tasks, the Encoder can be flexibly constructed, such as by an LSTM [275] in a task of machine translation (MT) or by a CNN [276] in a task of computer vision (CV). Since it could largely determine the precision of learned embedding in Z as well as the lower bound of reconstruction error between X and Y, appropriately constructing the Encoder is an important step. In (d), diverse strategies can be adopted in corrupting X to $\tilde{X}$, like Additive isotropic Gaussian noise (GS), Masking noise (MN) and Salt-and-pepper noise (SP), among which SP is only for binary data.

Seq2seq with attention mechanism aims to distinguish the (generally different) importance of z_* by learning their respective attention weights a_*, in order to pick up the most representative ones which can carry most of X's information and to represent them with an embedding as output. This way enables the learned embedding to maximally represent the primary information of X with a limited dimension. Methodologically, attention weights can be learned by an attention function mapping a query Q and a set of key-value pairs K, V to an output (Fig. 8 gives details). Subsequently, Kim et al. [290] proposed a structured attention network (SAN), which can take the structured dependency of the attention layer into account. By substituting a self-attention component used to learn the attention weights for the RNN structure in the Encoder, Transformer [291] can support parallel computing (Fig. 8 gives details). The attention and self-attention mechanisms can also be applied to the AutoEncoder framework-based general graph embedding. For instance, GATs [292] can construct a masked self-attention block layer for graph convolution (Fig. 8 gives details).

4.2. Recommendation involving side information

This section retrospects recommendation models involving side information from two different perspectives: technique-oriented and scenario-oriented.

Firstly, in technique, extracting properties of users and items from general graphs and incorporating them into recommendation can be implemented by the three categories of techniques retrospected in Sec. 4.1. Based on the translation-based general graph embedding framework, for instance, TransRec [293] can generalize TransE to sequential recommendation, in a manner that regarding a user as the translation between the pairs of items he

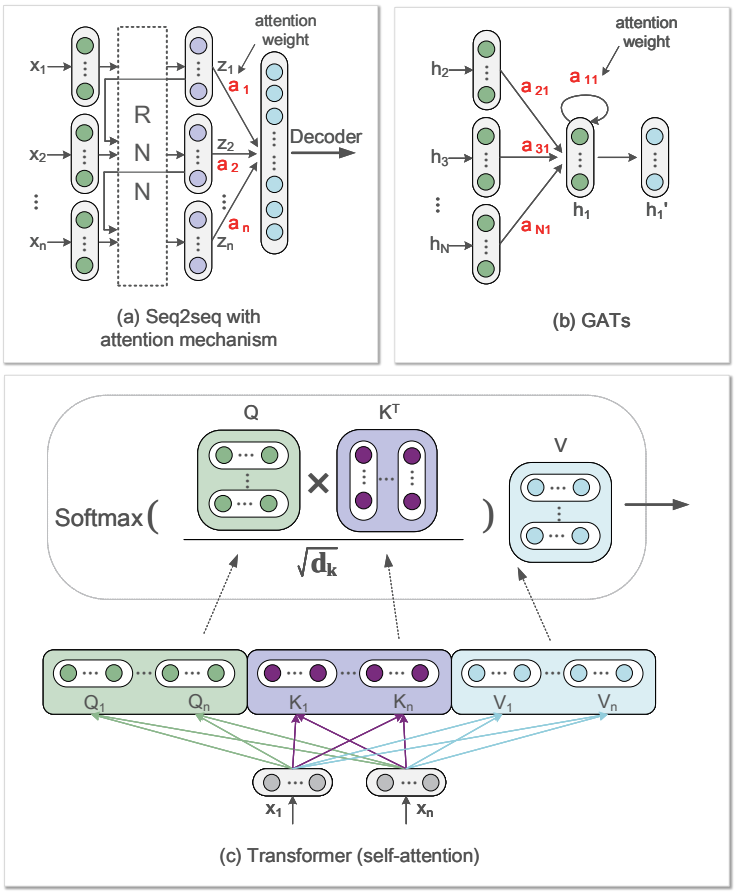

Figure 8: Schematics of Seq2seq with attention mechanism, Transformer(self-attention) and GATs. In (a), as an Encoder, Seq2seq can learn the embedding Z with a fixed dimension compressed from $z_1, ..., z_n$ (generated from the input $x_1, ..., x_n$) by minimizing $p(y_1, ..., y_{T'}|x_1, ..., x_T) = \prod_{t=1}^{T'} p(y_t|Z, y_1, ..., y_{t-1})$, where $X = (x_1, .., x_T)$ is the input, $y_1, ..., y_{T'}$ is the output, and T is generally not equal to T'. After incorporating the attention mechanism into Seq2seq, attention weights can be learned by $a \sim p(a|K, Q)$, where K represents the input X and Q is the result of the last iteration in the Decoder. Different from the pure Seq2seq where the output Z is directly compressed from $z_1, ..., z_n$, in Seq2seq with attention mechanism the importance of z_* can be assigned by attention weights a_*. In (b), as for a node i, GATs takes the node's neighborhood $h_* \in \mathbb{R}^F$ as the input, based on which the attention weights between node i and its neighbors can be learned by $a_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^{\mathrm{T}}[Wh_i||Wh_j]))}{\sum_{k \in N_i} \exp(\text{LeakyReLU}(\mathbf{a}^{\mathrm{T}}[Wh_i||Wh_k]))}$, where $W \in \mathbb{R}^{F' \times F}$ is a shared weight matrix, $a \in \mathbb{R}^{2F'}$ is a weight vector, N_i is some neighborhood of node i, and $||$ is the concatenation operator. After that, the embedding of the input $h'_i \in \mathbb{R}^{F'}$ can be learned by $h_i = \sigma(\sum_{j \in N_i} a_{ij} W h_j)$. In (c), the input $X = (x_1, ..., x_n)$ is firstly mapped into Q_*, K_* and V_* through transformation matrices W^Q, W^K and W^V, respectively, as model parameters. Based on them, $z_1, ..., z_n$ can be generated by $Z = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$, where d_k is the dimension of Q_* and K_*. Q, K and V are all vectors, among which K is equal to V.

sequentially interacted during a certain period (of course these item pairs are adjacent in the timeline in terms of this user's interactions), which means that as for a user embedding $\overrightarrow{\text{user}}$ the relation $\overrightarrow{\text{prev. item}} + \overrightarrow{\text{user}} \approx \overrightarrow{\text{next. item}}$ should always hold between the embedding of the user's previous interacted item $\overrightarrow{\text{prev. item}}$ and that of his next (or sequential) interacted items $\overrightarrow{\text{next. item}}$. However, TransRec inherits the defect of TransE that only be suited for 1-to-1 relations (*i.e.*, take different relational types as the same one). In practice, since user's next interacted items are usually diverse in types, distinguishing the diverse relational types between these item pairs is supposed to be a necessity. By building multiple semantic-specific matrices for the transition between distinctive item types, CTransRec [294] can further represent the different relational types between item pairs, extending TransRec to the 1-to-n relational translation.

In order to employ the meta path-based general graph embedding framework into recommendation, a common-used strategy is to base a user-item proximity matrix on random walks, which will be used as the input of the matrix factorization framework to implement recommendation. Take HIN [295] for instance. In a general graph, HIN hypothesizes that a user's preferences for items could be visualized by random walking along designed meta paths from this user to items within a boundary centered on this user. Under this assumption, the accumulated frequency of passes through an item on the user's random walks could be recognized as the item's proximity with the user, which could further be used to quantify the user's preference for this item. Intuitively, the higher the accumulated frequency is, the more indirect relations (*i.e.*, paths) between the user-item pair, so the more possible that an unobserved interaction exists between them. After implementing the diffusion mechanism on each user, a global user-item proximity matrix can be built, based on which the matrix factorization will run for recommendation. Similarly, by generating multiple local user-item proximity matrices corresponding to different meta paths, FMG [296] can learn multiple distributed embeddings for each user-item pair. These embeddings will be used as the input of FM [153], which enables FMG to represent the feature interactions between inter-meta paths. In the practice of various recommendation scenarios, under the meta path-based framework, designing efficient meta paths which could maximally capture the diverse characteristics of a general graph to rule random walking directly determines a model's recommendation accuracy. At this point, for instance, by remaining only the nodes of users and items in meta paths, Shi et al. [297] designed a graph schema used to distinguish the two nodal types during random walking. However, manually designing meta paths could require much expert knowledge. In order to automatically generate meta paths by random walking without hand-craft design, by triggering multiple "ripples" from user's historical interacted items as centers, RippleNet [298] can stimulate user's preference diffusion in a general graph along hierarchical propagation traces as multiple meta paths. Besides, by using an LSTM layer [275] to identify the holistic semantic of meta paths, KPRN [299] can realize reasoning on meta paths. Chen et al. [300] further took into account the temporal factors in general graphs and proposed a temporal meta path guided explainable recommendation (TMER).

When it comes to applying the deep learning-based general graph embedding framework into recommendation, the AutoEncoder framework motivated a variety of recommendation models. For instance, according to explicit user-item interactions (*i.e.*, ratings, where the unobserved ones are represented by value 0), AutoRec [301] firstly builds a rating vector with a dimension equaling the number of all items. Then, this rating vector will be used as the input of AutoEncoder, which can complete the missing elements (*i.e.*, ratings 0) in the input as predicted ratings by reconstructing it. In terms of recommendation based on implicit user-item interactions, by assuming that user's preferences for items should be represented with the corrupted layer of SDAE because of the possible incompleteness of one's observed implicit interactions, Wu et al. [302] applied SDAE to recommendation. Besides, Liang et al. [303] firstly applied variational AutoEncoders (VAEs) [282] to recommendation. As a Bayesian version of VAEs, CVAE [304] can simultaneously employ explicit user-item interactions as well as item's profiles into recommendation. Similar in being equipped with the attention and self-attention mechanisms to the AutoEncoder framework, recommendation models can also be incorporated with them, For instance, by means of the attention mechanism, AFM [305] can weigh the feature interactions in FM [153]. Xu et al. [306] proposed a multi-layered long- and short-term self-attention network (LSSA) for sequential recommendation.

It is positive that the above thoughts of employing general graph embedding techniques into recommendation have also made successful attempts in the industry. For instance, by taking the records about user's session-based online activities in Taobao as side information, Alibaba [307] constructed a weighted and directed network that is comprised of user-item interactions in a continuous period [308] and that can be used to extract hidden consumption habits of users to promote recommendation accuracy. By taking the semantic networks, social networks and profile networks of users as side information simultaneously, SHINE [309] can predict the sign of a sentiment link (*i.e.*, each of user's attitudes towards items) without analyzing textual information like user's comments on items.

On the other hand, from a scenario-oriented perspective, the rest of this section retrospects the utilization of user's social information [310–312] and location information [313] as side information in recommendation, in which case recent years have pioneered the location-based social recommendation [314] thriving for a long period. In general, user's online information can be collected from online social platforms where users are allowed to express their needs, desires or attitudes towards items and events by such as tweeting or posting. This online information usually carries user's preferences for items, which is more timely and explicit compared with user-item interactions [99]. Collecting user's online information across multiple social platforms could link these different platforms together by their shared users. In other words, diverse side information could be therefore integrated in this way, which contributes to improving recommendation accuracy by collaboratively revealing more about user's preferences. In this regard, it is not to deny that users are encouraged to use their social platform accounts (like Twitter or Weibo)

to log on to other online platforms like Amazon or Taobao, which enables user's comments on products, a type of social information revealing one's hobbies or demands shared on social platforms, to be employed in e-commerce recommender systems.

For the purpose of incorporating user's online social information in recommendation, multitudinous recommendation models were proposed. For instance, by collecting user's properties from microblogging, METIS [99] can learn item's demographics which could be depicted with the properties of its interacted users. Methodologically, by means of classification algorithms METIS can detect user's purchase intents and at the same time can extract item's demographic attributes from user's microblogging information in real-time, based on which recommendation can be run by matching users and items. Take MART [315] for another instance. Built on user's online social information, MART firstly constructs several social networks (*i.e.*, a type of general graphs) related to recommender systems by users as bridges. When coming to a cold-start user, MART can retrieve the user's social information from these social networks, which would provide valuable information for recommendation such as his attributes like gender, age or career, aiming to analyze his preferences for items. The rationale behind MART lies in learning a transformation matrix used to map user's information of properties from social networks to his corresponding embeddings in recommender systems. Apart from user's properties, there is more valuable information from social networks that could be employed in recommendation. Another instance is user's social ties [316], a concept used to distinguish the strong and weak ties of users' friendship. With respect to it, Granovetter et al. [317] exemplified the more crucial role of weak ties in social networks compared with strong ones, because the weak ties can largely determine the connection patterns of clustered components in social networks. Wang et al. [318] discovered that by distinguishing user's social ties in a social network the recommendation accuracy can be improved. Based on the discovery, by measuring the tie strength between node pairs with Jaccard's coefficient to classify them into strong or weak ones, Wang et al. proposed TBPR, a recommendation model that can distinguish each user's interacted items into five types in order to quantify user's different preferences for them. Finally, these distinguished preferences will be the input of the BPR framework [190], used to perform the pair-wise ranking strategy for training.

Besides, methods of employing user's location information in recommendation are also a recent research focus. For instance, Wang et al. [319] discovered that more than 80% of a user's visited new places are located within a 10km vicinity of the user's latest visited places, unraveling the influence of one's location information on his decisions of future visits. The resources of user's location information are diverse, among which user's spatial trajectory information is a general one collected from user's mobile techniques, like GPS [320], WiFi [321] or ad-hoc networks [322] with permission. Besides, user's check-in information of visits is also a source. In contrast with spatial trajectory information, it requires lower privacy rights because it can be directly collected from user's visit records. However, it cannot be updated in real-time as the spatial trajectory information does. In fact, a user's location information is usually related to his online social information, for one would like to share profiles of his visited places on social platforms, such as comments, experiences, photos or moods. To some extent these profiles can be used to express the motivation of one's visits to some places [323] and further be used to uncover his preferences for them. In this regard, recent research has explored enhancing the recommendation accuracy by employing user's location information as a supplement to online social information, called the location-based social recommendation.

For the purpose of incorporating user's location information in recommendation, models of location-based social recommendation are proposed. For instance, by means of user's check-in records to link users and locations, LFBCA [319] constructs a bipartite graph, based on which the proximity between user-location pairs can be measured by analyzing users' co-visits under the UBCF framework. In addition, based on PageRank with BCA [324], LFBCA can also utilize user's social relationships in uncovering (hidden) relations between users in the bipartite graph, in order to enhance the precision of proximity measurement. However, two flaws of LFBCA were clear. In the first place, ignoring the timeliness of user's location information could cause embarrassment, like recommending a place that is far away from one's current location. To deal with it, users can be subdivided into two categories: in-town and out-town [325]. Second, user's location information is usually sparse because of one's limited visits to new places due to the constraints of his economic or time income. For that, digging out more side information of places, like their profiles given by users or local people's preferences for them [326], as a supplement is helpful. Among them, place profiles are particularly valuable because they can reveal sentimental attributes, like user's positive or negative attitudes towards these places, which can be used to infer whether a place meets a user's sentimental preferences and needs. For instance, sentimental attributes can be used to measure the proximity between user-location or location-location pairs [323], in which case the techniques of textual sentiment analysis [327, 328] in NLP could wield its predominance. In recent years, taking into account temporal factors in the location-based social recommendation is setting off a new tendency [329, 330].

4.3. Summary

Fig. 9 concisely summarizes the key developments of general graph embedding techniques (*i.e.*, by the inner timeline in the figure) and recommendation models based on these techniques (*i.e.*, by the outer timeline of the figure), respectively. Apparently, all the three categories of general graph embedding techniques can be applied in recommendation, in which case those based on meta path and deep learning are the broadest ones. Relevant research in recent years has concentrated more on taking into account the temporal factors involved in a general graph and also more on representing the multiplicity of heterogeneous graphs.

Integrating general graphs with user-item bipartite graphs by linking their co-existed nodes is a direct strategy for providing the bedrock of employing general graph embedding techniques in recommendation. However, this strategy could dilute the significance of users and items as a result of the diversity of nodal types in a general graph. It seems that resorting to the attention and self-attention mechanisms as means to highlight the two most salient nodal types (*i.e.*, users and items) in recommendation is a promising focus.

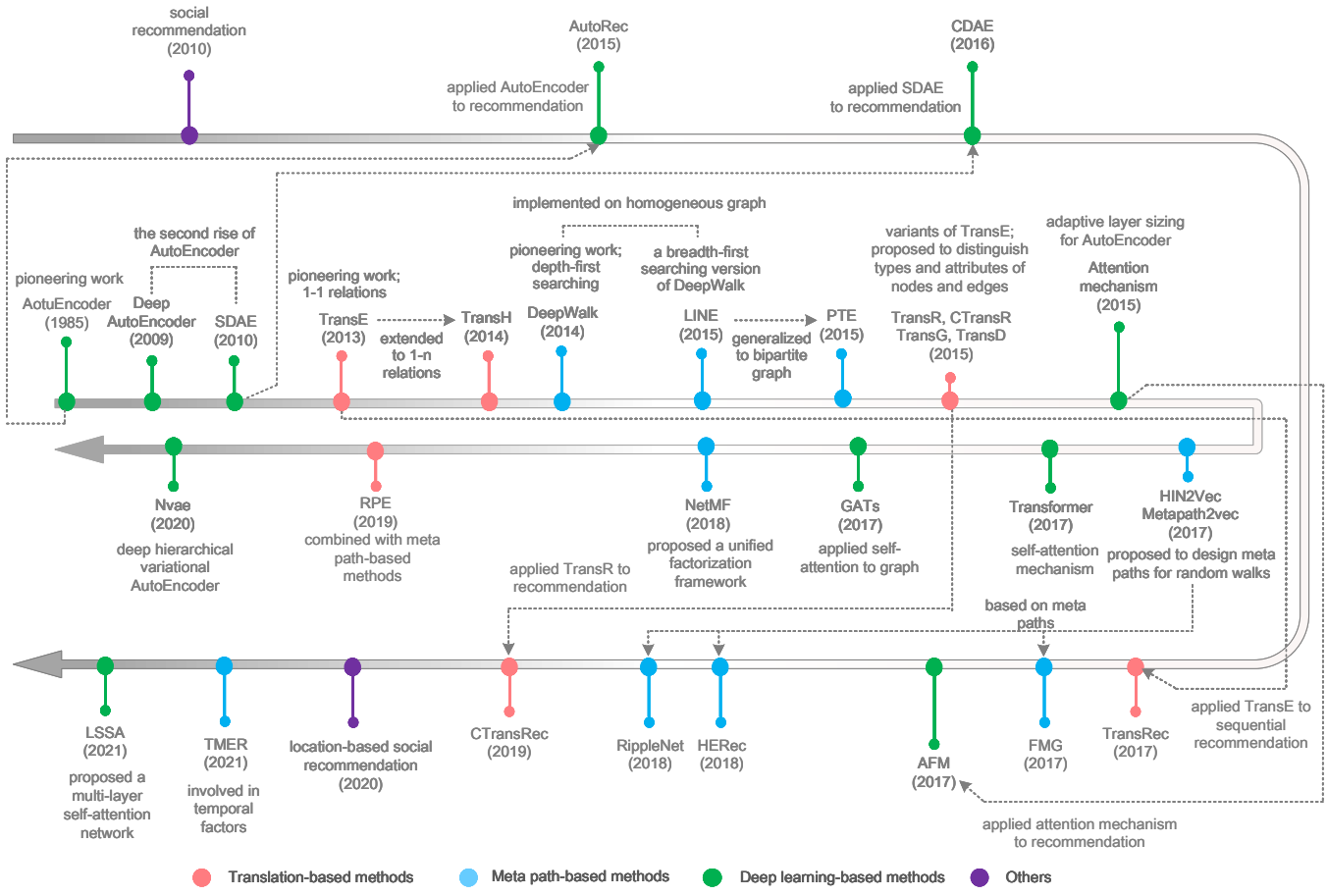

Figure 9: Timeline of key developments in general graph embedding for recommendation.

5. Knowledge graph embedding for recommendation

As illustrated in Sec. 2.1.2, knowledge graphs are usually recognized as the most complex case of general graphs. The complexity of knowledge graphs is characterized by an extreme abundance of node types and edge types, making knowledge graph embedding confront several challenges, like the large-scale, multiplicity and evolution of knowledge graphs. On the other hand, benefit from their complexity property of far more diverse carried information

compared to general graphs, knowledge graphs have a virtue of uncovering abundant hidden user-item relations dear to researchers, in order to enhance the recommendation accuracy.

Feasible as general graph embedding techniques are in employing knowledge graphs in recommendation, the scalability of them is constrained by the large-scale of knowledge graphs. Defects are not limited to this, general graph embedding techniques are usually insufficient in capturing and preserving the features in a high multiplicity of knowledge graphs. Faced with this situation, one way out of the dilemma was to develop more efficient techniques for knowledge graph embedding. For that, in the first place, Sec. 5.1 illustrates the three challenges of knowledge graph embedding and retrospects the corresponding solutions (*i.e.*, techniques for knowledge graph embedding). Based on that, Sec. 5.2 retrospects the application of these techniques in recommendation from two perspectives: embedding-based and path-based. Finally, Sec. 5.2 highlights two promising research directions.

5.1. Three challenges of knowledge graph embedding

In general, knowledge graph embedding meets three challenges [331]: the large-scale, multiplicity and evolution of knowledge graphs. The large-scale challenge means that knowledge graphs are usually too enormous in scale to be completely observed, making general graph embedding techniques have difficulty in running with an acceptable computing complexity. Besides, the multiplicity of knowledge graphs is generally characterized by the following three cases: graphs with multi-typed nodes and single-typed edges, graphs with single-typed nodes and multi-typed edges or graphs with both multi-typed nodes and edges. In view of these cases, sufficiently capturing and preserving the multiplicity is firmly anchored in precisely representing a knowledge graph. For that purpose, general graph embedding techniques would prefer to be used to represent knowledge graphs but are constrained from doing so by the lack of versatile expert knowledge in designing meta paths for such complex knowledge graphs and the general inability of translation-based techniques in capturing higher-order topological features. Moreover, since knowledge is constantly and rapidly growing in scale and multiplicity, the evolution of knowledge graphs largely challenges the computing complexity in updating the embeddings once learned from knowledge graphs.

In the face of the large-scale challenge, graph neural networks (GNNs) [25, 332, 333] can provide an efficient framework supporting parallel computing and fast response. The rationale behind GNNs lies in label propagation (or message passing) [334, 335], which is built on the assumption that each node with its features can be reconstructed by means of its connection with neighbors as well as its neighbors' features in a graph. Methodologically, in GNNs each node consists of two components: aggregator and updater. By collecting and aggregating the features from a node's neighbors, the aggregator can build a node's context embedding. After that, the node's context embedding combined with other input information (like side information) will be used to generate its embedding by the updater. Furthermore, by stacking each node's K different adjacent neighbors or repeating the propagation process K times on each node, the receptive field of GNNs can be expanded to a K-hop graph neighborhood. Subsequently, under the GNNs framework, more variants were proposed, among them transductive learning [336] and inductive learning [337, 338] are two prevalent strategies for implementing the label propagation, where the former aims to infer the labels of unlabelled nodes based on labeled ones and the latter aims to learn a global function for labeling. In detail, for example, the inductive learning strategy takes each node in a graph as both the receiver and disseminator of information (like features) from and to its neighbors within specified hops, in which case the features of nodes can be updated during such a continuous information exchanging process, a parallel computing process of label propagation used to refine the embeddings of all nodes until which reaching a global convergence. Moreover, by extracting a graph's critical components based on the spectral sparsification theory [339], spectral methods [265, 340] can achieve a fast response. By generating new relationships between hidden semantic properties (such as analogical properties [341] or circular correlation [342]) to equip a knowledge graph with enriched topological properties, generative graph methods [286, 341, 342] can speed up the learning process on large-scale graphs.

When it comes to capturing and preserving the multiplicity of knowledge graphs, GNNs have become adept in representing both the features and topological structures of knowledge graphs, in a manner of the label propagation between nodes. Because GNNs have come to see the merits of constructing highly expressive embeddings by composing extracted multi-scale localized spatial features like convolutional neural network (CNN) [343] dose while being suitable for non-Euclidean data like graphs. In addition, in terms of building other effective graph representations which could carry the multiplicity of a graph as much as possible, on which embedding techniques run, multi-viewed graphs [344] can provide a clearer view of depicting a heterogeneous graph containing single-typed nodes and multi-typed edges (Fig. 10 gives examples). Based on them, multi-viewed graph embedding aims to integrate the information carried by the different views of a multi-view graph together (termed as collaboration [345]) and at the same time to preserve their distinctive properties carried by each view (termed as preservation [345]). In this way, the learned embeddings can represent both the local features of nodes in each homogeneous community

(*i.e.*, views) as well as their global features across different communities. Methodologically, multi-viewed clustering [346–348] and multi-viewed matrix factorization [217, 349] were two attempts while still lacking sufficient collaboration as a consequence of their simple and independent concatenations of the learned embeddings from different views. In fact, completing both the collaboration and preservation is inextricably tied up with the good performance of multi-viewed graph embedding. For that purpose, more efficient methods have been proposed recently. For instance, by building random walk pairs across different views, Mvn2Vec [345] can capture the global structure of a graph. In addition, for each node i, by learning an overall-shared embedding $\boldsymbol{h}_i^c \in \mathbb{R}^d$ of it as well as its distinctive embeddings $\boldsymbol{h}_i^v \in \mathbb{R}^s$ in a specific view v, MNE [350] can combine $\boldsymbol{h_i^c}$ and $\boldsymbol{h_i^v}$ by $\boldsymbol{h}_i^v = \boldsymbol{h}_i^c + \omega^v \boldsymbol{X}^{vT} \boldsymbol{h}_i^v$, where $\boldsymbol{X}^{vT} \in \mathbb{R}^{s \times d}$ is a transformation matrix and ω^v indicates the global importance of view v, in order to complete the preservation and collaboration for node i. Among these works, apparently, distributing proper importance weights ω^v to different views is essential [351], in which case the attention mechanism seems to be promising assistance.

As for the other case of multiplicity that graphs with both multi-typed nodes and edges, multi-layered graphs [352–354] provide another category of intuitive graph representations, which have diverse applications ranging from multi-scale graph embedding [355, 356] to cross-domain scenarios like critical infrastructure systems [357] or collaboration platforms [358]. Among them, the simplest form of multi-layered graphs is coupled heterogeneous graph, which consists of two types of nodes and three types of edges (Fig. 10 gives examples). When it comes to a knowledge graph about actors and movies, the two layers in Fig. 10 can be constructed by an actor community representing the cooperation among actors and a movie community representing the category relations between movies, respectively. The edges between the two layers can be used to represent the participation relationships between actors and movies. Methodologically, a variety of translation-based methods have been proposed recently for coupled heterogeneous graph embedding. For instance, as a node-oriented model, EOE [359] can map nodal embeddings from different layers to each other through an overall-shared harmonious matrix. However, such a node-oriented strategy could neglect the influences of inter-connection edges between different layers. To deal with that, an edge-oriented strategy [272, 360] was proposed to take into account the inter-connection edges. When coming to a multi-layered graph with enormous layers, applying meta path-based methods can achieve better computing efficiency. For instance, by building ruled random walks across different layers, PMNE [361] can capture the global topological features of a coupled heterogeneous graph. Furthermore, based on the previous example of an actor-movie knowledge graph, if the attributes of actors and movies need to be represented with nodes in multi-types in layers, a more complex multi-layered graph that can distinguish different types of nodes and edges is required, as shown in Fig. 10 for example. For that, promising models include GATNE-T/GATNE-I [362] and DMNE [363], to name a few. One common idea of these models is to apply fast learning algorithms to represent the heterogeneous information in each layer efficiently and use an attention mechanism to learn their different importance.

Capturing and preserving the temporal factors involved in dynamical knowledge graphs in the face of the evolution challenge has become a research direction in recent years. Benefitting from its structure and label propagation (or message passing) mechanism, GNNs can be used to represent dynamic graphs [333]. Other attempts include ctRBM [364], M^{2}DNE [365], HTNE [366] and TGAT [367], to name a few. See [368] for an in-depth review.

5.2. Recommendation involving knowledge

Embedding-based and path-based are two typical strategies for equipping the knowledge graph embedding techniques retrospected in Sec. 5.1 in recommender systems, in order to efficiently employ knowledge in recommendation for higher accuracy and better interpretability.

By learning the supplemental features as embeddings of users and items from knowledge graphs, the embedding-based strategy can employ knowledge in recommendation by methods of cross-domain recommendation or transfer learning retrospected in Sec. 3.1.4. Built on this strategy, various models were proposed. For instance, by means of a textual knowledge graph, kaHFM [369] can learn item's embeddings of semantic features as a supplement to factorization machine (FM) [153]. After extracting the structural, textual and visual features of items from three different knowledge graphs, CKE [370] can construct a synthetic embedding for each item as a supplement. In general, since the efficiency in representing the features of entities and relations in knowledge graphs, it is a common-used method to apply GNNs to the embedding-based strategy [335, 371]. Among them, as for a user, how to accurately measure the high-order proximity between his interacted items and those far away from him (named "remote" items) in a knowledge graph is an open and important issue because the "remote" items can be recognized as candidates of diverse recommendations to the user. For such purposes, for instance, by deepening the range of neighborhood from one hop to multiple hops away, KGCN [335] can measure the proximity between nodes with a longer distance. Meanwhile, discriminating the unequal influences on a node from its different neighbors is another focus. Based on GATs [292], KGAT [371] builds an attention-based aggregator that can learn the different

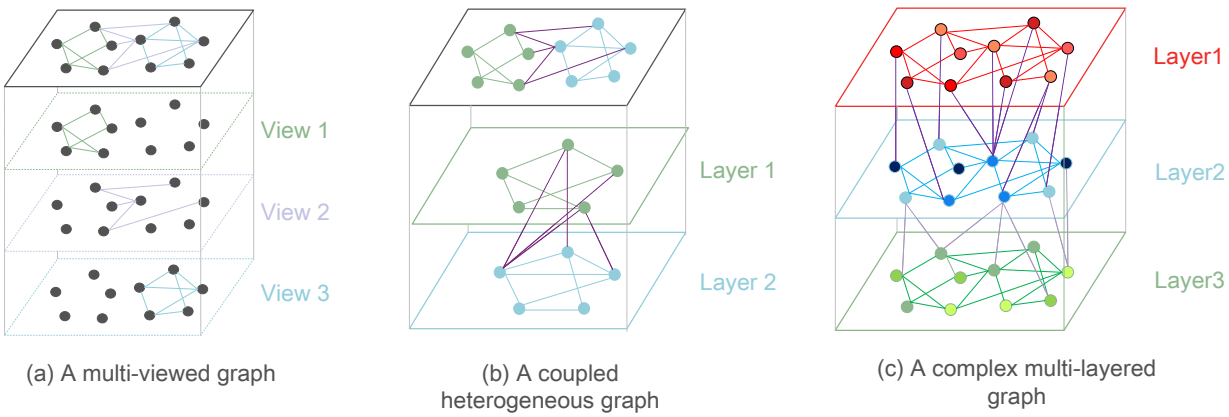

Figure 10: Examples of a multi-viewed graph, a coupled heterogeneous graph and a complex multi-layered graphs. In (a), the multi-viewed graph represents a heterogeneous graph containing single-typed nodes colored in grey and three types of edges colored in green, purple and blue, respectively, where each view represents a homogeneous component connected by edges of the same type. In (b), the multi-layered graph represents a heterogeneous graph containing two types of nodes and three types of edges by clustering the nodes into two layers colored in green and blue, respectively, where each layer contains nodes of the same type and the interconnections between layers are colored in purple. In (c), it is a complex multi-layered heterogeneous graph with cross-domains, where the intra-connections in each layer are one-typed that belong to a particular domain while the nodes are heterogeneous.

contribution weights from a node's neighbors. In the same way, in the hyperbolic space Hyper-Know [372] builds an aggregator that can learn hyperbolic attention with Einstein midpoint. In the training process of these models, unobserved edges were used to be commonly considered as negative samples [86, 212]. However, Togashi et al. [373] uncovered that such a negative sampling strategy could aggravate the cold start problem as a consequence of taking new items as negative samples, leading to biased and sub-optimal recommendation results toward already popularized items. To make up for the flaw, by applying pseudo-labeling [374] to distinguish possible weak-positive pairs in unobserved edges, Togashi et al. proposed KGPL. Besides, more subsequent works [375, 376] have been devoted to this focus in recent years.

In contrast to the embedding-based strategy which aims to learn supplementary features as embeddings of users and items from knowledge graphs, the rationale for path-based strategy lies in extracting user-item relations from knowledge graphs through random walking across multiple types of nodes and edges guided by designed meta paths, in order to achieve explainable recommendation [45, 48–50]. By following the extracted user-item relations along multi-hop meta paths, user's preferences for items could be better understood [299]. Intuitively, take Fig. 2 for example, by following the relations (or paths) *(Alice, Watched, Avengers4)∧(Avengers4, ProduceBy, TWDC)∧(TWDC, Produce, Avengers2)* and *(Alice, Watched, Avengers4)∧(Avengers4, StarBy, Robert)∧(Robert, star, Avengers 2)*, which shows that *Avengers 2* was produced by the same company as that of *Avengers 4* and also has a common actor in *Avengers 4* that Alice watched in the past, it is deducible that *Avengers 2: age of ultron* would meet Alice's preferences for movies. Apparently, this strategy could equip recommendation with abilities in reasoning and explainability. Methodologically, in order to precisely generate random walks on a knowledge graph which are integrated with a bipartite graph containing user nodes and item nodes, the path-based strategy generally meets the following three issues: first, how to design reasonable meta paths used to rule random walking? Second, how to measure user-item proximity through random walks? Three, how to distinguish the different importance of multiple random walks in order to highlight the most valuable ones? As resorts, for instance, by learning the embeddings of nodes in each random walk, KPRN [299] can represent the hidden sequential patterns between nodes, which will be used to learn the random walk's embedding representing its holistic semantics through a long short-term memory (LSTM) [275]. These learned embeddings of random walks can be used to measure user-item proximity. Furthermore, by means of a weighted pooling layer, KPRN can also distinguish the different importance of multiple random walks between a user-item pair. However, the pooling layer is built by enumerating all possible random walks between a user-item pair, requiring a high computing complexity, particularly on knowledge graphs with an enormous scale. Alternatively, Xian et al. [377] designed a soft reward strategy that can quickly highlight the most valuable random walks between a user-item pair. After taking into account user's history click sequences,

Zhu et al. [72] further proposed a strategy for weighting random walks. Recently, Chen et al. [300] discovered that ignoring the temporal factors in user-item interactions could result in less convincing and inaccurate recommendation explainability. For that, Chen et al. proposed TMER to construct item-item instance paths between consecutive items for sequential recommendation.

In application, there are two promising research directions of knowledge graph embedding-based recommendation. One is the conversational recommender system (CRS) [70, 378–380], which is equipped with abilities in chatting with users, understanding their expressions (*i.e.*, language) and reasoning their preferences for items on the advice of semantic knowledge or domain knowledge. The motivation of CRS lies in taking actives steps to solve the issue of user privacy infringement encountered by conventional recommendation systems, like those that manage to collect side information from user's social and location information, which is generally protected as privacy. In contrast, CRS can straightforwardly ask needed information for recommendation from users, in which case users have endowed the rights to determine which information they would prefer to share. In general, CRS consists of two components [381]: one is the dialog component designed to interpret human language into proper machine-readable forms and generate responses to users through a multi-round natural language conversational system [382, 383]. The other is the recommendation component designed to reason user's preferences for items by analyzing the content of user-machine conversations, based on which recommendations can be generated and returned to users. Based on the two components, the implementation of CRS basically includes the following three stages: initiation, conversation and display [384]. In detail, by raising a suitable question the dialog component can initiate a conversation with users. This conversation will continue to collect needed information for recommendation from users through multi-round questions and answers. After analyzing user's preferences for items from the collected information, the recommendation component can pick up possible candidates which would meet user's interest and then generate the top-N items as recommendations returned to users. The three stages will iteratively implement many rounds till the returned recommendations satisfy users. In addition, by means of knowledge graphs CRS can also reason on user's preferences for items, realizing the explainability of recommendation results. For example, suppose a user expressed his affection on *Avengers 2: age of ultron* and *Avengers 4: the final battle* in conversations with CRS. Combined with the knowledge graph in Fig. 2, CRS can infer that the user may also like other movies produced by TWDC or stared by Robert. Actually, by building user's profiles in the dialog component and matching them with item's attributes (like reviews [381]) in the recommendation component as illustrated above, CRS can be recognized as a content-based recommendation method. Methodologically, item-oriented method [384–389] is the most common-used implementation for CRS, which aims to uncover hidden relations between items by employing knowledge graphs. Furthermore, Zhou et al. [390] discovered that the word-level enrichment in conversations could reveal user's personal habits in word usage and hence would be valuable to understand user's preferences. In light of that, Zhou et al. proposed a word-oriented method, which can assist the item-oriented models in learning user embeddings. Recently, research into efficiently guiding users from non-recommendation scenarios to some desired ones via proactive conversations has become a crucial direction of CRS [381, 391–393].

The other promising direction of knowledge graph embedding-based recommendation in the application is news recommender systems [394–397]. Compared with recommendation based on non-textual scenarios, since the primary information for recommendation in news recommender systems is textual content [398], there raise three more questions: first, the news is distinctly time-sensitive. In other words, the relevance to reality makes the news fade away rapidly over time, where out-of-date news is constantly replaced by new ones. Second, user's interests in the news are generally topic-sensitive, diversified and changeable, related to current social focuses and issues. Third, news language is generally comprised of both professional knowledge structured logically and common sense presented causally. To understand, reason and represent it, techniques in NLP combined with graph embedding are both required, which are more sophisticated than previous ones for non-textual recommendation. Equipping content-based recommendation methods with knowledge as external resources enable the learning and representation of relatedness between news words as well as their latent knowledge-level connections more precisely. For instance, by employing an enormous knowledge graph in recommendation DKN [398] can build relations between the word entities identified and extracted from the news, used to construct a small domain knowledge graph for each piece of news consisting of these word entities and relations. Based on the domain knowledge graph, DKN can learn the embeddings of news pieces through a convolutional neural network (CNNs) [399].

5.3. Summary

Fig. 11 concisely summarizes the key developments of knowledge graph embedding techniques (*i.e.*, by the inner timeline in the figure) and recommendation models based on these techniques (*i.e.*, by the outer timeline in the figure), respectively. Conceivably, compared with the developments of bipartite graph-based and general graph

embedding-based recommendation shown in Figs. 6 and 9, the developments of knowledge graph embedding-based one began, for the first time in large numbers, in recent years. its developments have advanced into research into efficiently incorporating multiplex and evolving large-scale knowledge in recommendation.

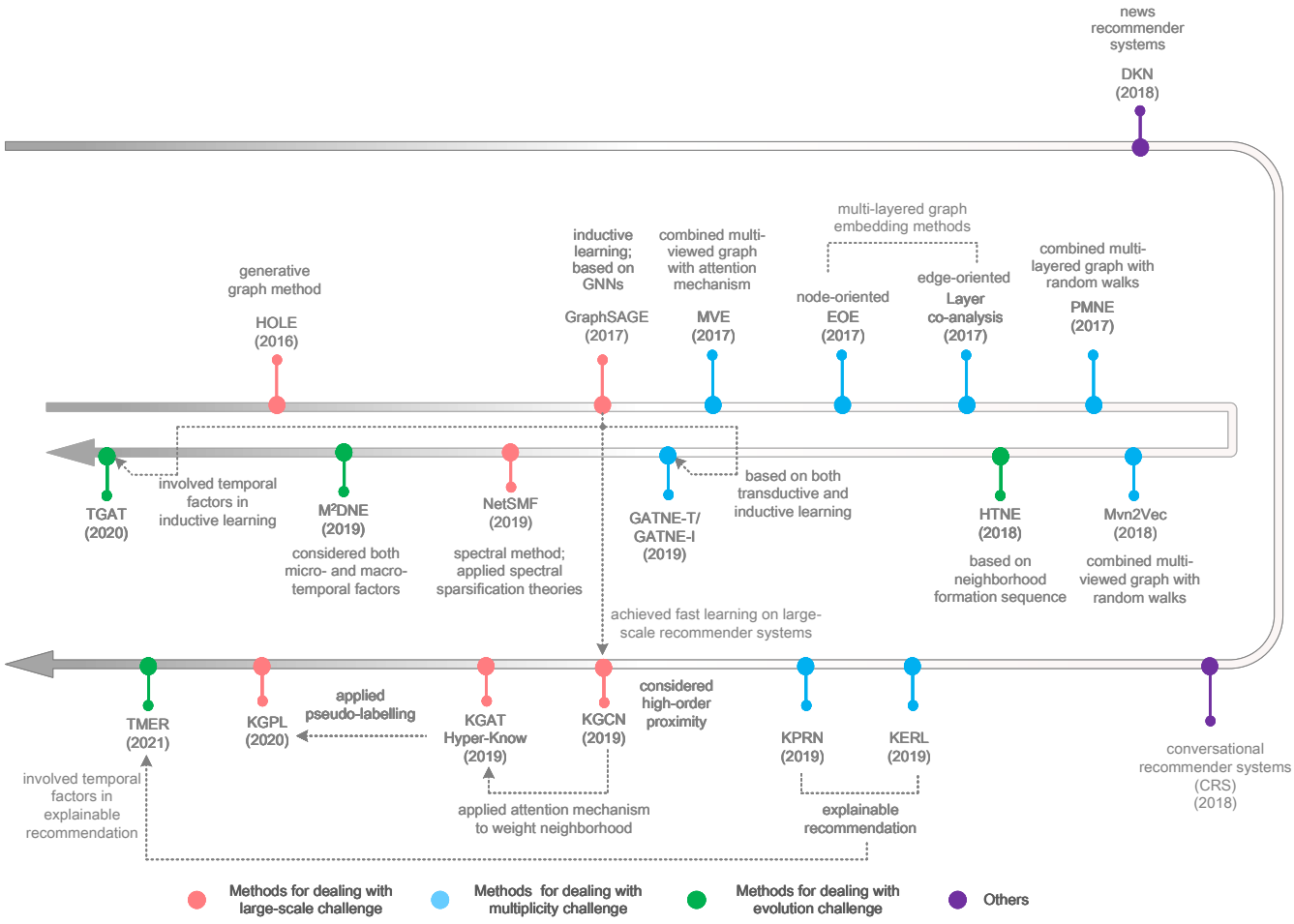

Figure 11: Timeline of key developments in knowledge graph embedding for recommendation.

6. Performance evaluation

This section designs and runs control experiments and displays experimental results on representative graph embedding-based recommendation models and the most common-used conventional recommendation models, comparing their pros and cons on six tasks related to different recommendation scenarios, data scales and data sparsity, in order to make a trade-off between them oriented to specific scenarios. In the first place, Sec. 6.1 designs the experiment setups, including data sets, evaluation metrics and evaluated models. Then, Sec. 6.2 presents experimental results and analysis.

6.1. Experiment setups

Although comprehensive evaluations on conventional recommendation models or graph embedding-based recommendation models did by existed research or published program libraries [400, 401], there still lacks sufficient comparisons between both of them under a unified experimental framework, providing little prospect on analyzing

and utilizing their respective strengths to complement each other. For that gap, this section designs six recommendation tasks for predicting both implicit and explicit user-item interactions, based on which several graph embedding-based and conventional recommendation models will be evaluated on five common-used metrics.

6.1.1. Data sets

Three data sets from two different recommendation scenarios and with distinctive data scales and sparsity used in experiments are described in Tab. 12. Among them, MovieLens 100K[1] and MovieLens 1M[2] are two data sets from a popular online movie recommender system named MovieLens where users are allowed to explicitly express their preferences for watched movies by giving ratings, which will be used to implement further movie recommendations. As a control experimental group, the two data sets record the ratings of movies given by users while being different from scales and sparsity, used to evaluate the performance of recommendation models in the same recommendation scenario but with different data scales and sparsity. In addition, another data set is Jester 617K[3] recording the ratings of jokes given by users, which is from a popular joke review platform where users are allowed to explicitly express their preferences for jokes.

Table 12: Descriptions of data sets. Density, defined as the ratio of the observed number of interactions to the maximum possible number of interactions between all users and items, is used to quantify data sparsity. $|U|$ and V denote the number of users and items, respectively.

Datasets	$\|U\|$	$\|I\|$	Observed interactions	Density(%)
MovieLens 100K	610	9724	100836	1.7
MovieLens 1M	6040	3682	1000209	4.497
Jester 617K	24938	100	616913	24.738

As explicit user-item interactions, the ratings recorded in the three data sets are slightly different in value: in MovieLens 100K, user i gave a rating of item j on a scale from 0.5 through 5, *i.e.*,$\mathrm{r}_{\mathrm{ij}}^{(100\mathrm{K})} \in [0.5, 5]$, with step 0.5, while in MovieLens 1M it is $\mathrm{r}_{\mathrm{ij}}^{(1\mathrm{M})} \in [1, 5]$ with step 1, and in Jester 617K it is $\mathrm{r}_{\mathrm{ij}}^{(617\mathrm{K})} \in [-10, 10]$ with step 0.0005. For unity, $\mathrm{r}_{\mathrm{ij}}^{(617\mathrm{K})} \in$ [-10, 10] is normalized to $\bar{\mathrm{r}}_{\mathrm{ij}}^{(617\mathrm{K})} \in [0, 5]$ by $\bar{\mathrm{r}}_{\mathrm{ij}}^{(617\mathrm{K})} = 5 \times \frac{\mathrm{r}_{\mathrm{ij}}^{(617\mathrm{K})}\text{-}(\text{-}10)}{10\text{-}(\text{-}10)}$ in experimental settings. Furthermore, by converting the ratings no less than 3 into value 1 while the rest of the ratings less than 3 into value 0, the corresponding implicit user-item interactions can be constructed. In this way, six different recommendation tasks are designed for evaluations of recommendation models.

Based on the above settings to implement model evaluations, each of the data sets is organized into tuples (user, item, rating/0/1) and is randomly split into a training set by 80% of the data set and a test set by 20% of the data set. Following that, recommendation models can implement based on the observed interactions in the training set, generating a recommendation list for each user. Then, the hit rate of the recommendation list can be calculated by checking how many items recommended to each user meet the observed interacted ones according to the test set. However, such a split strategy inevitably yields deviations of the recommendation accuracy based on different realizations (*i.e.*, different randomly split training-test sets). To assure the reliability of results, 20 realizations are randomly generated, based on which each of the six recommendation tasks repeatedly and independently implement in order to obtain the average and standard deviation of results.

6.1.2. Evaluation metrics

Evaluation metrics [402, 403] used in experiments include *mean absolute error (MAE)* [404] and *root mean squared error (RMSE)* [296, 404] for evaluations on recommendation accuracy in predicting explicit user-item interactions, and *Precision* [319], *Recall* [319, 326] and *Rank* [405] for those in predicting implicit user-item interactions.

Specifically, denote the training sets and test sets as $\mathcal{T}_{\mathrm{r}}$ and $\mathcal{T}_{\mathrm{e}}$, respectively, in which only the users and items that appear both in the training sets and test sets remain while the ones not satisfying the condition as well as the corresponding interactions between them are removed. Denote the observed rating on item j by user i in the test

[1] https://grouplens.org/datasets/movielens/
[2] https://grouplens.org/datasets/movielens/
[3] http://eigentaste.berkeley.edu/dataset/

set as r_{ij} and the predicted one as $\hat{r}_{ij}$. Then, MAE and RMSE are defined as

$$\text{MAE} = \frac{1}{|\mathcal{T}_\text{e}|} \sum_{(i,j)\in\mathcal{T}_\text{e}} |r_{ij} - \hat{r}_{ij}|, \tag{6.1}$$

$$\text{RMSE} = \left(\frac{1}{|\mathcal{T}_\text{e}|} \sum_{(i,j)\in\mathcal{T}_\text{e}} (r_{ij} - \hat{r}_{ij})^2 \right)^{\frac{1}{2}}. \tag{6.2}$$

Intuitively, MAE and RMSE can quantify the overall deviation between r_{ij} and $\hat{r}_{ij}$, where a smaller value indicates a better recommendation accuracy.

For a user i, denote the generated recommendation list containing the top L predicted items in which the user might be most likely to be interested as $\mathcal{O}_i$, and denote the items that the user has interacted with within the test set as $\mathcal{S}_i$. Suppose there are M users in the test set. Then Precision and Recall are defined as

$$\text{Precision} = \frac{\sum_{i=1}^{M} |\mathcal{O}_i \bigcap \mathcal{S}_i|}{ML}, \tag{6.3}$$

$$\text{Recall} = \frac{\sum_{i=1}^{M} |\mathcal{O}_i \bigcap \mathcal{S}_i|}{|\mathcal{T}_\text{e}|}, \tag{6.4}$$

where $L = 50$ in experiments. Intuitively, Precision and Recall can quantify the hit rate of a recommendation list (*i.e.*, in a recommendation list the proportion of items meeting a user's interacted items within the test set) in two different methods. A higher Precision or Recall indicates a better recommendation accuracy.

Rank is a more fine-grain metric to evaluate the recommendation accuracy. For a user i, denote the Rank of each of his interacted item $j \in \mathcal{S}_i$ by $\text{R}_{\text{ij}} = \frac{1}{\log_2(p_j + 2)}$ if item j appears in $\mathcal{O}_i$, where p_j is the ranking position of item j in $\mathcal{O}_i$ that ranging from 0 to L, and 0 otherwise. Then, the overall Rank of items in $\mathcal{T}_\text{e}$ is defined as

$$\text{Rank} = \frac{1}{|\mathcal{T}_\text{e}|} \sum_{(i,j)\in\mathcal{T}_\text{e}} \text{R}_{\text{ij}}\,. \tag{6.5}$$

A higher overall Rank indicates a better recommendation accuracy.

6.1.3. Evaluated models

The representative graph embedding-based recommendation models pioneering subsequent variants in each category from Secs. 3, 4 and 5 are partially selected as evaluated models in experiments. As benchmarks, some of the most commonly used conventional recommendation models are selected. Tab. 13 gives an overview of them.

Table 13: Evaluated models. Corresponding to the data sets, recommendation tasks in experiments are divided into two aspects: predicting explicit and implicit user-item interactions, respectively.

Recommendation tasks	Graph embedding-based models	Conventional models (benchmarks)
Predicting explicit user-item interactions	FunkSVD, PMF, FM, AutoRec	Overall Average, UserKNN, ItemKNN
Predicting implicit user-item interactions	GMF, MLP, NCF, TransE	UserKNN, ItemKNN, HybridKNN HeatS, ProbS, HybridS

In detail, as for recommendation tasks for predicting explicit user-item interactions, three graph embedding-based recommendation models are selected from Secs. 3.1.1, 3.1.2 and 4.1.3, including FunkSVD [78], probabilistic matrix factorization (PMF) [181] and AutoRec [301], respectively. In addition, factorization machine (FM) [153] selected from Sec.3.1.1 is a representative model used to tackle the sparsity problem in recommendation. Meanwhile, as benchmarks, three conventional recommendation models including Overall Average, user-based k-nearest neighbor collaborative filtering (UserKNN) [406] and item-based k-nearest neighbor collaborative filtering (ItemKNN) [28] are selected, among which Overall Average [13] takes a user's average rating on items in the training set as the predicted rating on his non-interacted items. UserKNN is illustrated in Sec. 2.1.4 and ItemKNN is an item-oriented version of UserKNN. HybridKNN is a combination of UserKNN and ItemKNN. The recommendation accuracy of these models in predicting explicit user-item interactions is evaluated by metrics MAE and RMSE.

As for recommendation tasks for predicting implicit user-item interactions, three graph embedding-based recommendation models are selected from Sec. 3.1.3, including generalized matrix factorization (GMF) [195], multi-layer perceptron (MLP) [195, 407] and neural collaborative filtering (NCF) [195]. In addition, another evaluated model is TransE [251] selected from Sec. 4.1.1, used to explore the performance of the application of models oriented to general graphs on bipartite graphs. Meanwhile, six conventional recommendation models, including UserKNN [406], ItemKNN [28], hybrid k-nearest neighbor collaborative filtering (HybridKNN) [408], heat spreading algorithm (HeatS) [29], probabilistic spreading algorithm (ProbS) [30] and hybrid spreading (HybridS) [409], are selected as benchmarks, among which HeatS, ProbS and HybridS are models established on theories of physical dynamics, and HybridS combines the advantages of ProbS oriented to high recommendation accuracy and HeatS oriented to high recommendation diversity [410]. The recommendation accuracy of these models in predicting implicit user-item interactions is evaluated by metrics Precision, Recall and Rank.

6.2. Results and analysis

On the basis of the above experiment setups, experimental results of evaluated models in predicting explicit and implicit user-item interactions are shown in Tabs. 14 and 15, respectively. The hyper-parameter settings in experiments of evaluated models are given in Tab. A1 in Appendix A. These hyper-parameter settings were determined by tuning on each of the first randomly split realization of the three data sets, respectively, where the hyper-parameter tuning results in experiments are presented with the tables in Appendix B. Codes and data are available[4].

6.2.1. Predicting explicit user-item interactions

In predicting explicit user-item interactions (*i.e.*, ratings), graph embedding-based recommendation models overall outperform conventional recommendation models in recommendation accuracy, in which case such the advantage of graph embedding-based models becomes more salient with the increase of data scales while data sparsity seems not to be a decisive factor. However, graph embedding-based models are overall less stable than conventional models.

Table 14: Results on explicit user-item interactions. All results to the nearest 0.001 are averaged over 20 realizations, and the value in brackets is the standard deviation to the nearest 0.001. To make the comparison clearer, the comparatively lower average MAE and average RMSE of models on each data set are in bold. Note that n.a. indicates that FunkSVD failed on Jester617K as a consequence of the gradient disappearance and explosion problem [411].

Data sets	**MovieLens 100K**		**MovieLens 1M**		**Jester 617K**	
Metrics	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
Overall Average	0.827 (0.005)	0.909 (0.003)	0.934 (0.001)	0.966 (0.001)	1.144 (0.031)	1.070 (0.015)
UserKNN	0.697 (0.004)	0.906 (0.005)	0.772 (0.001)	0.977 (0.001)	0.960 (0.024)	1.185 (0.030)
ItemKNN	0.717 (0.004)	0.906 (0.005)	0.719 (0.001)	0.908 (0.001)	0.915 (0.021)	1.131 (0.027)
Avg.	0.747 (0.003)	**0.907 (0.004)**	0.808 (0.001)	0.950 (0.001)	1.006 (0.025)	**1.129 (0.024)**
FunkSVD	0.705 (0.005)	0.918 (0.006)	0.693 (0.002)	0.887 (0.002)	n.a.	n.a.
PMF	0.740 (0.006)	0.969 (0.008)	0.694 (0.006)	0.882 (0.007)	0.897 (0.019)	1.132 (0.025)
FM	0.678 (0.009)	0.881 (0.006)	0.682 (0.005)	0.870 (0.003)	0.892 (0.019)	1.123 (0.027)
AutoRec	0.742 (0.006)	0.959 (0.007)	0.724 (0.003)	0.919 (0.004)	1.053 (0.046)	1.304 (0.067)
Avg.	**0.716 (0.007)**	0.932 (0.007)	**0.698 (0.004)**	**0.890 (0.004)**	**0.947 (0.028)**	1.186 (0.119)

As shown in Tab. 14, the average MAE of graph embedding-based recommendation models on MovieLens 100K, Jester 617K and MovieLens 1M, which are sorted by data scales in ascending order, are 4.15% less, 5.86% less and 13.61% less than that of conventional recommendation models, respectively. At the same time, in terms of the average RMSE, it is 2.76% greater, 5.05% greater and 6.32% less than that of conventional recommendation models, respectively. Apparently, with the increase of data scales, the average MAE and RMSE of graph embedding-based recommendation models are getting overall lower than those of conventional ones. It appears that the machine learning methodology adopted by graph embedding-based recommendation models plays a large role in those experimental results, benefiting from the better data fitting performance based on a larger data scale. However, when concerning data sparsity the results in Tab. 14 seem to show nothing of its role. Although the increase of density from MovieLens 100K to MovieLens 1M brings overall greater advantages in MAE and RMSE of graph

[4] https://github.com/pitteryue/Recommender-Systems

embedding-based models beyond conventional models, it is the opposite as the density continues to increase from MovieLens 1M to Jester 617K. Admittedly, these experimental results are insufficient to prove the existence of an optimal data sparsity making the overall recommendation accuracy of graph embedding-based recommendation models better than that of conventional ones. On the other hand, Tab. 14 shows that graph embedding-based recommendation models have greater values of standard deviations of the average MAE and average RMSE than those of conventional one on all three data sets, manifesting the overall lower stability of graph embedding-based recommendation models compared with conventional ones.

In addition, it can be observed in Tab. 14 that FM outperforms all other models in recommendation accuracy on the three data sets. It appears that the rationale for modeling all interactions between variables (*i.e.*, feature intersection) behind FM helps uncover richer patterns hidden in sparse data for a more accurate recommendation, particularly on MovieLens 100K with the lowest data density. This observation encourages the research into models like FM that can base on small and sparse data sets to achieve high enough recommendation accuracy. Similar to few-shot learning [412, 413], FM and its variants might be a potential direction for the future breakthrough.

Table 15: Results on implicit user-item interactions. The length of recommendation list is set to be 50 (denoted by @50). All results to the nearest 0.001 are averaged over 20 realizations, and the value in brackets is the standard deviation to the nearest 0.001. To make the comparison clearer, the comparatively higher average Precision, average Recall and average Rank of models on each data set are in bold. Note that as for conventional models HeatS merely oriented to recommendation diversity is not taken into account for the average results of recommendation accuracy.

MovieLens 100K			
Metrics	**Precision@50**	**Recall@50**	**Rank@50**
UserKNN	0.126 (0.002)	0.247 (0.003)	0.078 (0.001)
ItemKNN	0.125 (0.002)	0.245 (0.003)	0.079 (0.001)
HybridKNN	0.141 (0.002)	0.274 (0.003)	0.089 (0.001)
HeatS	0.013 (0.001)	0.024 (0.001)	0.017 (0.051)
ProbS	0.120 (0.002)	0.235 (0.003)	0.075 (0.001)
HybridS	0.128 (0.002)	0.251 (0.004)	0.080 (0.001)
Avg.	**0.128 (0.002)**	**0.250 (0.003)**	**0.080 (0.001)**
GMF	0.099 (0.003)	0.192 (0.005)	0.057 (0.002)
MLP	0.072 (0.004)	0.140 (0.009)	0.040 (0.003)
NCF	0.117 (0.006)	0.228 (0.011)	0.072 (0.004)
TransE	0.055 (0.002)	0.107 (0.004)	0.032 (0.001)
Avg.	0.086 (0.004)	0.167 (0.007)	0.050 (0.003)
MovieLens 1M			
Metrics	**Precision@50**	**Recall@50**	**Rank@50**
UserKNN	0.155 (0.000)	0.280 (0.001)	0.086 (0.000)
ItemKNN	0.157 (0.000)	0.283 (0.001)	0.086 (0.000)
HybridKNN	0.178 (0.000)	0.322 (0.001)	0.100 (0.000)
HeatS	0.058 (0.002)	0.104 (0.004)	0.021 (0.001)
ProbS	0.127 (0.001)	0.230 (0.001)	0.072 (0.000)
HybridS	0.143 (0.000)	0.259 (0.001)	0.082 (0.000)
Avg.	**0.152 (0.000)**	**0.275 (0.001)**	**0.085 (0.000)**
GMF	0.137 (0.004)	0.248 (0.007)	0.071 (0.003)
MLP	0.100 (0.014)	0.181 (0.026)	0.041 (0.008)
NCF	0.120 (0.015)	0.216 (0.027)	0.050 (0.009)
TransE	0.050 (0.006)	0.089 (0.011)	0.024 (0.003)
Avg.	0.102 (0.010)	0.184 (0.018)	0.047 (0.006)
Jester 617K			
Metrics	**Precision@50**	**Recall@50**	**Rank@50**
UserKNN	0.034 (0.001)	0.755 (0.020)	0.331 (0.005)
ItemKNN	0.040 (0.000)	0.906 (0.002)	0.383 (0.001)
HybridKNN	0.041 (0.000)	0.913 (0.002)	0.389 (0.001)
HeatS	0.037 (0.000)	0.836 (0.003)	0.288 (0.004)
ProbS	0.042 (0.000)	0.948 (0.001)	0.414 (0.001)
HybridS	0.042 (0.000)	0.948 (0.001)	0.419 (0.001)
Avg.	**0.040 (0.000)**	**0.894 (0.005)**	**0.387 (0.002)**
GMF	0.039 (0.006)	0.879 (0.143)	0.332 (0.081)
MLP	0.042 (0.001)	0.939 (0.021)	0.358 (0.013)
NCF	0.042 (0.000)	0.946 (0.002)	0.365 (0.015)
TransE	0.031 (0.002)	0.705 (0.046)	0.215 (0.025)
Avg.	0.039 (0.002)	0.867 (0.053)	0.318 (0.036)

6.2.2. Predicting implicit user-item interactions

In predicting implicit user-item interactions (*i.e.*, 0/1), graph embedding-based recommendation models overall underperform conventional ones in recommendation accuracy on the three data sets, in which case recommendation scenario seems to be a decisive factor. Meanwhile, the lower stability of graph embedding-based recommendation models than conventional ones as manifested in Sec. 6.2.1 is also embodied in these tasks.

As shown in Tab. 15, graph embedding-based recommendation models underperform conventional ones on MovieLens 100K by 32.81%, 33.20% and 37.50% over average Precision@50, average Recall@50 and average Rank@50, respectively. On MovieLens 1M these results of graph embedding-based recommendation models are 32.89% less, 33.09% less and 44.71% less than those of conventional ones, respectively. On Jester 617K are 2.50% less, 3.02% less and 17.83% less, respectively. As these results put it, neither data scale nor sparsity is a decisive factor in recommendation accuracy. In other words, increasing data scale no longer brings consistent improvement in recommendation accuracy for graph embedding-based recommendation models. From Jester 617K to MovieLens 1M, the average Precision, average Recall and average Rank of graph embedding-based recommendation models are lower than those of conventional ones in a wider gap, while the case from MovieLens 100K to Jester 617K is the opposite. Meanwhile, it can be observed that decreasing data sparsity can shorten the accuracy gap between graph embedding-based recommendation models and conventional ones in terms of their average results on MovieLens 1M and Jester 617K, while this case does not hold on MovieLens 100K and MovieLens 1M. In view of that, the decisive factor influencing recommendation accuracy most likely lies in recommendation scenario distinguished by the difference in the relative number of users and items, for that of Jester 617K, a joke recommendation scenario dominated by the number of users, is distinguished from the cases on MovieLens 100K and 1M about a movie recommendation scenario. When concerning model stability, apparently, conventional recommendation models outperform graph embedding-based ones in terms of the standard deviation of their average results on all three data sets.

6.3. Summary

In conclusion, based on the above experimental results, this section provides some constructive suggestions on making a trade-off between graph embedding-based and conventional recommendation in different recommendation tasks, and also proposes some open questions for future research.

The graph embedding-based and conventional recommendation perform distinctively on different tasks. First, as for recommendation accuracy, graph embedding-based recommendation is a prior choice in predicting explicit user-item interactions, especially when the data scale is large. Meanwhile, in predicting implicit user-item interactions, conventional recommendation could still maintain a priority to graph embedding-based recommendation without considering the utilization of side information and knowledge. Second, when concerning stability, conventional recommendation is always a prior choice benefit from its less adjustment for hyper-parameters compared with graph embedding-based recommendation. Third, recommendation efficiency is also a necessity to be taken into account, especially for practical applications with big data. In this regard, graph embedding-based recommendation appears to be always considered as a priority because of its rationale for reusing the embeddings once learned for recommendation.

In practice, recommender systems usually involve both tasks for predicting explicit and implicit user-item interactions. Besides, recommender systems intrinsically develop from small to large in data scale and from sparse to dense in data sparsity, especially after being incorporated with side information and knowledge. Correspondingly, making a trade-off between graph embedding-based and conventional recommendation appears to vary in these different development stages of recommender systems, as the experimental results manifest in this section. As suggestions, in recommender systems' early developing stages conventional recommendation could perform well on a small data scale. At the same time its better explainability compared with graph embedding-based recommendation provides clearer views on user's behaviors, consider as straightforward feedback for guiding a more adaptive recommendation (models') configuration. With the increase of data scale when advancing into the following developing stages of recommender systems, graph embedding-based recommendation is supposed to gradually dominate recommendation configuration for better efficiency and accuracy. In the long run, in order to complement each other with their respective advantages, a versatile recommendation configuration mixing conventional and graph embedding-based recommendation models is an optimal strategy for improving recommender systems.

However, there still remain a few open questions. First, does data sparsity determines a model's recommendation performance? If it does, how? Second, how to select appropriate models for a specific recommendation scenario? Third, when designing a recommendation model, is it better to adopt a task-oriented strategy or a generalization-oriented strategy?

7. Discussions and outlook

The rapid development of computing resources and machine learning methodology contributes to the prevalence of graph embedding-based recommendation. In summary, as retrospected in this article, bipartite graph embedding-based recommendation can provide an extensible framework based on which the auxiliary information and temporal factors involved in user-item interactions can be incorporated. However, it can not sufficiently and efficiently incorporate side information or knowledge. To fill this gap, general graph embedding-based recommendation was invented. Furthermore, in order to enhance the abilities of recommendation models in representing the multiplicity and evolution of large-scale data, knowledge graph embedding-based recommendation has recently attracted many scholar attention. Nevertheless, among these techniques, flaws and challenges also exist as illustrated in this article. For that, complementing them to each other with their respective advantages is seemly promising, by such as combining embedding-based methods with path-based methods or extending shallow models to deep learning ones like deep matrix factorization [414]. To the end, this section puts forward some open questions of graph embedding-based recommendation as well as their correspondingly possible solutions as follows.

7.1. Graph topological analysis for recommendation

To promote model accuracy, how could graph topological analysis contribute to graph embedding-based recommendation? In making great strides in recommender systems from conventional recommendation to graph embedding-based one, are we really making much progress on its performance? This debate has been raised many times in recent years, where Dacrema et al. [62] threw cold water on DNN-based recommendation (a recently prevalent focus) and proved that conventional recommendation models can still achieve higher accuracy compared to graph embedding-based ones in some tasks, as verified in Sec. 6. In view of that, graph topological analysis (the rationale behind conventional recommendation) seemly still plays a large role in model's accuracy. As yet, recent research into combining graph topological analysis on subgraphs, motifs or neighborhoods with graph embedding-based recommendation has come to see the merit in improving the performance of model's accuracy. For future research, analyzing graph's higher-order topological characteristics [415–418], namely network cycles, cliques and cavities, and employing them into graph embedding-based recommendation appears to be a potential direction, for these characteristics can be used to uncover higher-order relations between nodes, which are beyond nodes' pair-wise relations where most current methods concentrated. Methodologically, in order to realize the combination between graph's higher-order characteristics and graph embedding-based recommendation, it could resort to utilizing higher-order graph topology as a guideline for random walking in some graph embedding-based recommendation models. In addition, it is also positive to base more sufficient graph representations (like hypergraphs [417, 419]) beyond the conventional three categories in Sec. 2.1.2 on their higher-order topology. Moreover, designing novel label propagation (or message passing) mechanisms for GNNs based on graph's higher-order topology is also a promising direction.

7.2. Explainable recommendation

Explainable recommendation drives recommendation models to evolve from machine perceptual learning based on data fitting to machine cognitive learning for reasoning, long regarded as a necessity for comprehending user's preferences for items, assuring recommendation results' reliability affecting user's trust (*i.e.*, users should know why did them receive these recommendations), and contributing to illuminating the black-box mechanism of graph embedding-based recommendation (at least in a phenomenon-level). For those purposes, promising future research directions might include knowledge-based explainable recommendation [45, 48–50], causal learning (causal inference) [51–57] and neural network interpretability [145].

7.3. Protect user's private information in recommendation

In practice, explicit interactions and side information (like social networks) are common-used information for recommendation. However, one might be loath to share with recommender systems the two kinds of recommendation since his privacy could be leaked or even be infringed by others. From another perspective, perceiving user's preferences for items (*i.e.*, explicit interactions) from the frequency of implicit user-item interactions (*e.g.*, click or play counts and viewing time) would be a potential solution. On the other hand, instead of escaping from utilizing this privacy-sensitive information, federated learning [145, 420, 421] is also a promising solution, which can require for user's private data to train several distributed models on user's local devices and then upload the parameters of these learned models to a terminal server to support an aggregated big model. In this process, user's privacy remains on their own devices without being shared with the terminal server. Nevertheless, since user's local devices

could lack enough computing efficiency for training a model, efficient embedding techniques for large-scale data are necessary for federated learning.

7.4. Employ social sciences and NLP techniques in recommendation

Sufficiently incorporating rich information in recommendation requires for identification and preservation of hidden patterns and multiplicity involved in this information by, for example, identifying different tie strengths in social networks or distinguishing possible abundant semantics of knowledge, in which case techniques of such as community detection or language representation and reasoning play essential roles. In light of that, social sciences and natural language processing (NLP) seemly are two fields most likely overlapped with research into graph embedding-based recommendation. As conversational recommender systems (CRS) has become a promising solution to user privacy protection and explainable recommendation, the techniques of CRS in semantic sentiment analysis and dialogue system are largely based on NLP.

In the future, it is conceivable that recommender systems will be a perpetual theme in the information age, as long as users have the demand to quickly access their preferred or needed items from millions of candidates and commercials have the willingness to efficiently reap profits by their products through Precise Marketing. The rise of graph embedding-based recommendation is just starting, underlying challenges are coming, and promising solutions are sprouting. Expanding the research into graph embedding-based recommendation from the current focus on recommendation accuracy to improving the diversity and novelty of recommendation could also broaden its potential applications. Opening up more applications of graph embedding-based recommendation in practice would give full play to its social and commercial values. It is worth noting that more comprehensive papers about recommender systems can be found and downloaded on public projects[5].

Acknowledgements

The author acknowledges Linyuan Lü, Shuqi Xu, Xu Na, Hao Wang and Honglei Zhang for their discussions and suggestions.

[5]https://github.com/hongleizhang/RSPapers

References

[1] L. Cai, Y. Zhu, The challenges of data quality and data quality assessment in the big data era, Data science journal 14 (2015).

[2] H. Fang, Z. Zhang, C. J. Wang, M. Daneshmand, C. Wang, H. Wang, A survey of big data research, IEEE network 29 (2015) 6–9.

[3] X. Zhao, X. Liu, X. Li, Tracking the spread of novel coronavirus (2019-ncov) based on big data, medRxiv (2020).

[4] S. H. Jain, M. Rosenblatt, J. Duke, Is big data the new frontier for academic-industry collaboration?, Jama 311 (2014) 2171–2172.

[5] A. Kumar, S. R. Sangwan, A. Nayyar, Multimedia social big data: Mining, in: Multimedia big data computing for IoT applications, Springer, 2020, pp. 289–321.

[6] D. Laney, et al., 3d data management: Controlling data volume, velocity and variety, META group research note 6 (2001) 1.

[7] J. Jacoby, Perspectives on information overload, Journal of consumer research 10 (1984) 432–435.

[8] K. Bontcheva, G. Gorrell, B. Wessels, Social media and information overload: Survey results, arXiv preprint arXiv:1306.0813 (2013).

[9] W. Hersh, Information retrieval, in: Biomedical Informatics, Springer, 2021, pp. 755–794.

[10] X. Wu, X. Zhu, G.-Q. Wu, W. Ding, Data mining with big data, IEEE transactions on knowledge and data engineering 26 (2013) 97–107.

[11] P. Resnick, H. R. Varian, Recommender systems, Communications of the ACM 40 (1997) 56–58.

[12] F. Ricci, L. Rokach, B. Shapira, Introduction to recommender systems handbook, in: Recommender systems handbook, Springer, 2011, pp. 1–35.

[13] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, T. Zhou, Recommender systems, Physics reports 519 (2012) 1–49.

[14] S. Xu, Q. Zhang, L. Lü, M. S. Mariani, Recommending investors for new startups by integrating network diffusion and investors' domain preference, Information Sciences 515 (2020) 103–115.

[15] Y. Himeur, A. Alsalemi, A. Al-Kababji, F. Bensaali, A. Amira, C. Sardianos, G. Dimitrakopoulos, I. Varlamis, A survey of recommender systems for energy efficiency in buildings: Principles, challenges and prospects, Information Fusion 72 (2021) 1–21.

[16] M. Singh, Scalability and sparsity issues in recommender datasets: a survey, Knowledge and Information Systems 62 (2020) 1–43.

[17] N. Idrissi, A. Zellou, A systematic literature review of sparsity issues in recommender systems, Social Network Analysis and Mining 10 (2020) 1–23.

[18] B. Lika, K. Kolomvatsos, S. Hadjiefthymiades, Facing the cold start problem in recommender systems, Expert Systems with Applications 41 (2014) 2065–2073.

[19] J. Gope, S. K. Jain, A survey on solving cold start problem in recommender systems, in: 2017 International Conference on Computing, Communication and Automation (ICCCA), IEEE, 2017, pp. 133–138.

[20] S. N. Gondaliya, K. Amin, Solutions to the cold start problem in recommender system: a survey, Journal of Software Engineering Tools & Technology Trends 5 (2019) 14–21.

[21] R. Sethi, M. Mehrotra, Cold start in recommender systems—a survey from domain perspective, in: Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2020, Springer Singapore, 2021, pp. 223–232.

[22] Z. Sun, Q. Guo, J. Yang, H. Fang, G. Guo, J. Zhang, R. Burke, Research commentary on recommendations with side information: A survey and research directions, Electronic Commerce Research and Applications 37 (2019) 100879.

[23] Q. Guo, Z. Sun, Y.-L. Theng, Exploiting side information for recommendation, in: International Conference on Web Engineering, Springer, 2019, pp. 569–573.

[24] G. Weikum, L. Dong, S. Razniewski, F. Suchanek, Machine knowledge: Creation and curation of comprehensive knowledge bases, arXiv preprint arXiv:2009.11564 (2020).

[25] Y. Gao, Y.-F. Li, Y. Lin, H. Gao, L. Khan, Deep learning on knowledge graph for recommender system: A survey, arXiv preprint arXiv:2004.00387 (2020).

[26] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, Q. He, A survey on knowledge graph-based recommender systems, IEEE Transactions on Knowledge and Data Engineering (2020).

[27] B. M. Sarwar, Sparsity, scalability, and distribution in recommender systems, University of Minnesota, 2001.

[28] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th international conference on World Wide Web, 2001, pp. 285–295.

[29] Y.-C. Zhang, M. Blattner, Y.-K. Yu, Heat conduction process on community networks as a recommendation model, Physical review letters 99 (2007) 154301.

[30] Y.-C. Zhang, M. Medo, J. Ren, T. Zhou, T. Li, F. Yang, Recommendation model based on opinion diffusion, EPL (Europhysics Letters) 80 (2007) 68003.

[31] P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowledge-Based Systems 151 (2018) 78–94.

[32] C. Yang, Z. Liu, C. Tu, C. Shi, M. Sun, Network embedding: Theories, methods, and applications, Synthesis Lectures on Artificial Intelligence and Machine Learning 15 (2021) 1–242.

[33] S. Bhagat, G. Cormode, S. Muthukrishnan, Node classification in social networks, in: Social network data analytics, Springer, 2011, pp. 115–148.

[34] A. M. Khattak, R. Batool, F. A. Satti, J. Hussain, W. A. Khan, A. M. Khan, B. Hayat, Tweets classification and sentiment analysis for personalized tweets recommendation, Complexity 2020 (2020).

[35] S. Verma, Z.-L. Zhang, Hunt for the unique, stable, sparse and fast feature learning on graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 87–97.

[36] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, S. Jaiswal, graph2vec: Learning distributed representations of graphs, arXiv preprint arXiv:1707.05005 (2017).

[37] A. Galland, M. Lelarge, Invariant embedding for graph classification, in: ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data, 2019.

[38] H. Chen, H. Koga, Gl2vec: Graph embedding enriched by line graphs with edge features, in: International Conference on Neural Information Processing, Springer, 2019, pp. 3–14.

[39] H. Wang, Y. Deng, L. Lü, G. Chen, Hyperparameter-free and explainable whole graph embedding, arXiv preprint arXiv:2108.02113 (2021).

[40] D. Liben-Nowell, J. Kleinberg, The link-prediction problem for social networks, Journal of the American society for information science and technology 58 (2007) 1019–1031.

[41] L. Lü, T. Zhou, Link prediction in complex networks: A survey, Physica A: statistical mechanics and its applications 390 (2011) 1150–1170.

[42] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, P. Merialdo, Knowledge graph embedding for link prediction: A comparative analysis, ACM Transactions on Knowledge Discovery from Data (TKDD) 15 (2021) 1–49.

[43] A. Kumar, S. S. Singh, K. Singh, B. Biswas, Link prediction techniques, applications, and performance: A survey, Physica A: Statistical Mechanics and its Applications 553 (2020) 124289.

[44] K. Wang, T. Zhang, T. Xue, Y. Lu, S.-G. Na, E-commerce personalized recommendation analysis by deeply-learned clustering, Journal of Visual Communication and Image Representation 71 (2020) 102735.

[45] Y. Zhang, X. Chen, Explainable recommendation: A survey and new perspectives, arXiv preprint arXiv:1804.11192 (2018).

[46] M. I. Jordan, T. M. Mitchell, Machine learning: Trends, perspectives, and prospects, Science 349 (2015) 255–260.

[47] S. Sun, Z. Cao, H. Zhu, J. Zhao, A survey of optimization methods from a machine learning perspective, IEEE transactions on cybernetics 50 (2019) 3668–3681.

[48] J. Gao, X. Wang, Y. Wang, X. Xie, Explainable recommendation through attentive multi-view learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019, pp. 3622–3629.

[49] M. PALMONARI, P. MINERVINI, Knowledge graph embeddings and explainable ai, Knowledge Graphs for Explainable Artificial Intelligence: Foundations, Applications and Challenges 47 (2020) 49.

[50] L. Xie, Z. Hu, X. Cai, W. Zhang, J. Chen, Explainable recommendation based on knowledge graph and multi-objective optimization, Complex & Intelligent Systems 7 (2021) 1241–1252.

[51] A. Gopnik, L. Schulz, L. E. Schulz, Causal learning: Psychology, philosophy, and computation, Oxford University Press, 2007.

[52] D. Liang, L. Charlin, D. M. Blei, Causal inference for recommendation, in: Causation: Foundation to Application, Workshop at UAI. AUAI, 2016.

[53] S. Bonner, F. Vasile, Causal embeddings for recommendation, in: Proceedings of the 12th ACM conference on recommender systems, 2018, pp. 104–112.

[54] L. Yao, Z. Chu, S. Li, Y. Li, J. Gao, A. Zhang, A survey on causal inference, arXiv preprint arXiv:2002.02770 (2020).

[55] Y. Wang, D. Liang, L. Charlin, D. M. Blei, Causal inference for recommender systems, in: Fourteenth ACM Conference on Recommender Systems, 2020, pp. 426–431.

[56] S. Xu, Y. Ge, Y. Li, Z. Fu, X. Chen, Y. Zhang, Causal collaborative filtering, arXiv preprint arXiv:2102.01868 (2021).

[57] S. Xu, Y. Li, S. Liu, Z. Fu, Y. Ge, X. Chen, Y. Zhang, Learning causal explanations for recommendation, in: The 1st International Workshop on Causality in Search and Recommendation, 2021.

[58] R. Duwairi, H. Ammari, An enhanced cbar algorithm for improving recommendation systems accuracy, Simulation Modelling Practice and Theory 60 (2016) 54–68.

[59] B. Yu, C. Zhou, C. Zhang, G. Wang, Y. Fan, A privacy-preserving multi-task framework for knowledge graph enhanced recommendation, IEEE Access 8 (2020) 115717–115727.

[60] Y. Zhang, Q. Ma, Doccit2vec: Citation recommendation via embedding of content and structural contexts, IEEE Access 8 (2020) 115865–115875.

[61] Y. Mao, S. A. Mokhov, S. P. Mudur, Application of knowledge graphs to provide side information for improved recommendation accuracy, arXiv preprint arXiv:2101.03054 (2021).

[62] M. F. Dacrema, P. Cremonesi, D. Jannach, Are we really making much progress? a worrying analysis of recent neural recommendation approaches, in: Proceedings of the 13th ACM Conference on Recommender Systems, 2019, pp. 101–109.

[63] J. Zhao, X. Wang, C. Shi, B. Hu, G. Song, Y. Ye, Heterogeneous graph structure learning for graph neural networks, in: 35th AAAI Conference on Artificial Intelligence (AAAI), 2021.

[64] S. Lim, J.-G. Lee, Motif-based embedding for graph clustering, Journal of Statistical Mechanics: Theory and Experiment 2016 (2016) 123401.

[65] H. Peng, J. Li, Q. Gong, Y. Ning, S. Wang, L. He, Motif-matching based subgraph-level attentional convolutional network for graph classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 5387–5394.

[66] S. Zhang, Z. Hu, A. Subramonian, Y. Sun, Motif-driven contrastive learning of graph representations, arXiv preprint arXiv:2012.12533 (2020).

[67] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 29, 2015.

[68] G. Ji, K. Liu, S. He, J. Zhao, Knowledge graph completion with adaptive sparse transfer matrix, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 30, 2016.

[69] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[70] Y. Sun, Y. Zhang, Conversational recommender system, in: The 41st international acm sigir conference on research and development in information retrieval, 2018, pp. 235–244.

[71] S. Okura, Y. Tagami, S. Ono, A. Tajima, Embedding-based news recommendation for millions of users, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 1933–1942.

[72] Q. Zhu, X. Zhou, J. Wu, J. Tan, L. Guo, A knowledge-aware attentional reasoning network for recommendation, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 6999–7006.

[73] Y. Dai, S. Wang, N. N. Xiong, W. Guo, A survey on knowledge graph embedding: Approaches, applications and benchmarks, Electronics 9 (2020) 750.

[74] S. Wang, L. Hu, Y. Wang, X. He, Q. Z. Sheng, M. Orgun, L. Cao, N. Wang, F. Ricci, P. S. Yu, Graph learning approaches to recommender systems: A review, arXiv preprint arXiv:2004.11718 (2020).

[75] L. Wu, X. He, X. Wang, K. Zhang, M. Wang, A survey on neural recommendation: From collaborative filtering to content and context enriched recommendation, arXiv preprint arXiv:2104.13030 (2021).

[76] S. Wang, L. Hu, Y. Wang, X. He, Q. Z. Sheng, M. A. Orgun, L. Cao, F. Ricci, P. S. Yu, Graph learning based recommender systems: A review, arXiv preprint arXiv:2105.06339 (2021).

[77] S. Choudhary, T. Luthra, A. Mittal, R. Singh, A survey of knowledge graph embedding and their applications, arXiv preprint arXiv:2107.07842 (2021).

[78] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37.

[79] G. Adomavicius, J. Bockstedt, S. Curley, J. Zhang, De-biasing user preference ratings in recommender systems, in: RecSys 2014 Workshop on Interfaces and Human Decision Making for Recommender Systems (IntRS 2014), 2014, pp. 2–9.

[80] N. Manjur, Exploring the diversity in the impact of colors of rating scales on user's rating behavior, Ph.D. thesis, University of Saskatchewan, 2021.

[81] K.-W. Park, B.-H. Kim, T.-S. Park, B.-T. Zhang, Uncovering response biases in recommendation, in: Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.

[82] C. Wang, Y. Zheng, J. Jiang, K. Ren, Toward privacy-preserving personalized recommendation services, Engineering 4 (2018) 21–28.

[83] W. Huang, B. Liu, H. Tang, Privacy protection for recommendation system: a survey, in: Journal of Physics: Conference Series, volume 1325, IOP Publishing, 2019, p. 012087.

[84] J. Liu, W. Pan, Z. Ming, Cofigan: Collaborative filtering by generative and discriminative training for one-class

recommendation, Knowledge-Based Systems 191 (2020) 105255.

[85] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: 2008 Eighth IEEE International Conference on Data Mining, Ieee, 2008, pp. 263–272.

[86] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, Q. Yang, One-class collaborative filtering, in: 2008 Eighth IEEE International Conference on Data Mining, IEEE, 2008, pp. 502–511.

[87] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, J. Sun, Temporal recommendation on graphs via long- and short-term preference fusion, in: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 723–732.

[88] J. Yu, T. Zhu, Combining long-term and short-term user interest for personalized hashtag recommendation, Frontiers of Computer Science 9 (2015) 608–622.

[89] Y. Wu, K. Li, G. Zhao, X. Qian, Long-and short-term preference learning for next poi recommendation, in: Proceedings of the 28th ACM international conference on information and knowledge management, 2019, pp. 2301–2304.

[90] H. Fang, D. Zhang, Y. Shu, G. Guo, Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations, ACM Transactions on Information Systems (TOIS) 39 (2020) 1–42.

[91] M. Xu, F. Liu, W. Xu, A survey on sequential recommendation, in: 2019 6th International Conference on Information Science and Control Engineering (ICISCE), IEEE, 2019, pp. 106–111.

[92] J. Chen, Q. Zhang, X. Huang, Incorporate group information to enhance network embedding, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 1901–1904.

[93] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, S. Wang, Learning graph-based poi embedding for location-based recommendation, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 15–24.

[94] L. Liao, X. He, H. Zhang, T.-S. Chua, Attributed social network embedding, IEEE Transactions on Knowledge and Data Engineering 30 (2018) 2257–2270.

[95] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, H. Liu, Attributed network embedding for learning in a dynamic environment, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 387–396.

[96] X. Huang, J. Li, X. Hu, Label informed attributed network embedding, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017, pp. 731–739.

[97] C.-J. Wang, T.-H. Wang, H.-W. Yang, B.-S. Chang, M.-F. Tsai, Ice: Item concept embedding via textual information, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 85–94.

[98] K. Mirkovski, Y. Jia, L. Liu, K. Chen, Understanding microblogging continuance intention, Information Technology & People (2018).

[99] X. W. Zhao, Y. Guo, Y. He, H. Jiang, Y. Wu, X. Li, We know what you want to buy: a demographic-based system for product recommendation on microblogs, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 1935–1944.

[100] R. Sun, X. Cao, Y. Zhao, J. Wan, K. Zhou, F. Zhang, Z. Wang, K. Zheng, Multi-modal knowledge graphs for recommender systems, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1405–1414.

[101] P. D. Taylor, T. Day, G. Wild, Evolution of cooperation in a finite homogeneous graph, Nature 447 (2007) 469–472.

[102] C. Shi, Y. Li, J. Zhang, Y. Sun, S. Y. Philip, A survey of heterogeneous information network analysis, IEEE Transactions on Knowledge and Data Engineering 29 (2016) 17–37.

[103] C. Shi, S. Y. Philip, Heterogeneous information network analysis and applications, Springer, 2017.

[104] D. A. Ferrucci, Introduction to "this is watson", IBM Journal of Research and Development 56 (2012) 1–1.

[105] X. L. Dong, Building a broad knowledge graph for products, in: 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, 2019, pp. 25–25.

[106] X. Luo, L. Liu, Y. Yang, L. Bo, Y. Cao, J. Wu, Q. Li, K. Yang, K. Q. Zhu, Alicoco: Alibaba e-commerce cognitive concept net, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 2020, pp. 313–327.

[107] F. M. Suchanek, G. Kasneci, G. Weikum, Yago: a core of semantic knowledge, in: Proceedings of the 16th international conference on World Wide Web, 2007, pp. 697–706.

[108] S. P. Ponzetto, M. Strube, Deriving a large scale taxonomy from wikipedia, in: AAAI, volume 7, 2007, pp. 1440–1445.

[109] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: The semantic web, Springer, 2007, pp. 722–735.

[110] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85.

[111] N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, S. Merugu, A web of concepts, in: Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2009, pp. 1–12.

[112] F. Suchanek, G. Weikum, Knowledge harvesting in the big-data era, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, 2013, pp. 933–938.

[113] G. Weikum, J. Hoffart, F. M. Suchanek, Ten years of knowledge harvesting: Lessons and challenges., IEEE Data Eng. Bull. 39 (2016) 41–50.

[114] G. Weikum, J. Hoffart, F. Suchanek, Knowledge harvesting: achievements and challenges, in: Computing and Software Science, Springer, 2019, pp. 217–235.

[115] C. Yang, M. Sun, Z. Liu, C. Tu, Fast network embedding enhancement via high order proximity approximation., in: IJCAI, 2017, pp. 3894–3900.

[116] J.-H. Yang, C.-M. Chen, C.-J. Wang, M.-F. Tsai, Hop-rec: high-order proximity for implicit recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 140–144.

[117] L. Leydesdorff, On the normalization and visualization of author co-citation data: Salton's cosine versus the jaccard index, Journal of the American Society for Information Science and Technology 59 (2008) 77–85.

[118] J. Lee Rodgers, W. A. Nicewander, Thirteen ways to look at the correlation coefficient, The American Statistician 42 (1988) 59–66.

[119] L. Hamers, et al., Similarity measures in scientometric research: The jaccard index versus salton's cosine formula., Information Processing and Management 25 (1989) 315–18.

[120] J. L. Herlocker, J. A. Konstan, J. Riedl, Explaining collaborative filtering recommendations, in: Proceedings of the 2000 ACM conference on Computer supported cooperative work, 2000, pp. 241–250.

[121] G. Linden, B. Smith, J. York, Amazon. com recommendations: Item-to-item collaborative filtering, IEEE Internet computing 7 (2003) 76–80.

[122] X. Su, T. M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in artificial intelligence 2009 (2009).

[123] Y. Koren, R. Bell, Advances in collaborative filtering, Recommender systems handbook (2015) 77–118.

[124] F. Yu, A. Zeng, S. Gillard, M. Medo, Network-based recommendation algorithms: A review, Physica A: Statistical Mechanics and its Applications 452 (2016) 192–208.

[125] X. Wang, Y. Liu, G. Zhang, F. Xiong, J. Lu, Diffusion-based recommendation with trust relations on tripartite graphs, Journal of Statistical Mechanics: Theory and Experiment 2017 (2017) 083405.

[126] M. J. Pazzani, D. Billsus, Content-based recommendation systems, in: The adaptive web, Springer, 2007, pp. 325–341.

[127] A. Van Den Oord, S. Dieleman, B. Schrauwen, Deep content-based music recommendation, in: Neural Information Processing Systems Conference (NIPS 2013), volume 26, Neural Information Processing Systems Foundation (NIPS), 2013.

[128] P. Lops, D. Jannach, C. Musto, T. Bogers, M. Koolen, Trends in content-based recommendation, User Modeling and User-Adapted Interaction 29 (2019) 239–249.

[129] H. Cai, V. W. Zheng, K. C.-C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, IEEE Transactions on Knowledge and Data Engineering 30 (2018) 1616–1637.

[130] F. Chen, Y.-C. Wang, B. Wang, C.-C. J. Kuo, Graph representation learning: A survey, APSIPA Transactions on Signal and Information Processing 9 (2020).

[131] C. D. Barros, M. R. Mendonça, A. B. Vieira, A. Ziviani, A survey on embedding dynamic graphs, arXiv preprint arXiv:2101.01229 (2021).

[132] S. Shalev-Shwartz, S. Ben-David, Understanding machine learning: From theory to algorithms, Cambridge university press, 2014.

[133] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al., Wide & deep learning for recommender systems, in: Proceedings of the 1st workshop on deep learning for recommender systems, 2016, pp. 7–10.

[134] Y. Zhang, R. Jin, Z.-H. Zhou, Understanding bag-of-words model: a statistical framework, International Journal of Machine Learning and Cybernetics 1 (2010) 43–52.

[135] T. Ogita, S. M. Rump, S. Oishi, Accurate sum and dot product, SIAM Journal on Scientific Computing 26 (2005) 1955–1988.

[136] S. Sra, S. Nowozin, S. J. Wright, Optimization for machine learning, Mit Press, 2012.

[137] N. Parikh, S. Boyd, Proximal algorithms, Foundations and Trends in optimization 1 (2014) 127–239.

[138] L. Bottou, Stochastic gradient learning in neural networks, Proceedings of Neuro-Nımes 91 (1991) 12.

[139] F. Niu, B. Recht, C. Ré, S. J. Wright, Hogwild!: A lock-free approach to parallelizing stochastic gradient descent,

arXiv preprint arXiv:1106.5730 (2011).

[140] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization., Journal of machine learning research 12 (2011).

[141] H. Li, Z. Lin, Accelerated proximal gradient methods for nonconvex programming, Advances in neural information processing systems 28 (2015) 379–387.

[142] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[143] S. Boyd, N. Parikh, E. Chu, Distributed optimization and statistical learning via the alternating direction method of multipliers, Now Publishers Inc, 2011.

[144] M. Crawshaw, Multi-task learning with deep neural networks: A survey, arXiv preprint arXiv:2009.09796 (2020).

[145] Y. Zhang, P. Tiňo, A. Leonardis, K. Tang, A survey on neural network interpretability, IEEE Transactions on Emerging Topics in Computational Intelligence (2021).

[146] G. W. Stewart, On the early history of the singular value decomposition, SIAM review 35 (1993) 551–566.

[147] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman, Indexing by latent semantic analysis, Journal of the American society for information science 41 (1990) 391–407.

[148] P. C. Hansen, Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank, SIAM Journal on Scientific and Statistical Computing 11 (1990) 503–518.

[149] G. Davis, S. Mallat, M. Avellaneda, Adaptive greedy approximations, Constructive approximation 13 (1997) 57–98.

[150] D. M. Hawkins, The problem of overfitting, Journal of chemical information and computer sciences 44 (2004) 1–12.

[151] J. S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, arXiv preprint arXiv:1301.7363 (2013).

[152] H. Ma, An experimental study on implicit social recommendation, in: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, 2013, pp. 73–82.

[153] S. Rendle, Factorization machines, in: 2010 IEEE International Conference on Data Mining, IEEE, 2010, pp. 995–1000.

[154] L. Hu, A. Sun, Y. Liu, Your neighbors affect your ratings: on geographical neighborhood influence to rating prediction, in: Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, 2014, pp. 345–354.

[155] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008, pp. 426–434.

[156] X. Ning, G. Karypis, Slim: Sparse linear methods for top-n recommender systems, in: 2011 IEEE 11th International Conference on Data Mining, IEEE, 2011, pp. 497–506.

[157] S. Kabbur, X. Ning, G. Karypis, Fism: factored item similarity models for top-n recommender systems, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013, pp. 659–667.

[158] D. D. Lee, H. S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (1999) 788–791.

[159] S. Zhang, W. Wang, J. Ford, F. Makedon, Learning from incomplete ratings using non-negative matrix factorization, in: Proceedings of the 2006 SIAM international conference on data mining, SIAM, 2006, pp. 549–553.

[160] Y. Xu, W. Yin, Z. Wen, Y. Zhang, An alternating direction algorithm for matrix completion with nonnegative factors, Frontiers of Mathematics in China 7 (2012) 365–384.

[161] A. Hernando, J. Bobadilla, F. Ortega, A non negative matrix factorization for collaborative filtering recommender systems based on a bayesian probabilistic model, Knowledge-Based Systems 97 (2016) 188–202.

[162] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, Q. Zhu, A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method, IEEE transactions on neural networks and learning systems 27 (2015) 579–592.

[163] O. Gouvert, T. Oberlin, C. Févotte, Ordinal non-negative matrix factorization for recommendation, in: International Conference on Machine Learning, PMLR, 2020, pp. 3680–3689.

[164] K. Liu, X. Li, Z. Zhu, L. Brand, H. Wang, Factor-bounded nonnegative matrix factorization, ACM Transactions on Knowledge Discovery from Data (TKDD) 15 (2021) 1–18.

[165] A. Tversky, I. Gati, Similarity, separability, and the triangle inequality., Psychological review 89 (1982) 123.

[166] P. Ram, A. G. Gray, Maximum inner-product search using cone trees, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 931–939.

[167] B. Kulis, et al., Metric learning: A survey, Foundations and trends in machine learning 5 (2012) 287–364.

[168] J. L. Suárez, S. García, F. Herrera, A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges, Neurocomputing 425 (2021) 300–322.

[169] S. Zhang, L. Yao, Y. Tay, X. Xu, X. Zhang, L. Zhu, Metric factorization: Recommendation beyond matrix factorization, arXiv preprint arXiv:1802.04606 (2018).

[170] C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, D. Estrin, Collaborative metric learning, in: Proceedings of the 26th international conference on world wide web, 2017, pp. 193–201.

[171] C. Ma, L. Ma, Y. Zhang, R. Tang, X. Liu, M. Coates, Probabilistic metric learning with adaptive margin for top-k recommendation, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1036–1044.

[172] I. Kraeva, G. Yakhyaeva, Application of the metric learning for security incident playbook recommendation, in: 2021 IEEE 22nd International Conference of Young Professionals in Electron Devices and Materials (EDM), IEEE, 2021, pp. 475–479.

[173] P. Li, A. Tuzhilin, Dual metric learning for effective and efficient cross-domain recommendations, IEEE Transactions on Knowledge and Data Engineering (2021).

[174] L. Vinh Tran, Y. Tay, S. Zhang, G. Cong, X. Li, Hyperml: a boosting metric learning approach in hyperbolic space for recommender systems, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 609–617.

[175] J. Lee, S. Kim, G. Lebanon, Y. Singer, Local low-rank matrix approximation, in: International conference on machine learning, PMLR, 2013, pp. 82–90.

[176] M. Aharon, M. Elad, A. Bruckstein, K-svd: An algorithm for designing overcomplete dictionaries for sparse representation, IEEE Transactions on signal processing 54 (2006) 4311–4322.

[177] N. Halko, P.-G. Martinsson, J. A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM review 53 (2011) 217–288.

[178] N. Srebro, T. Jaakkola, Weighted low-rank approximations, in: Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 720–727.

[179] X. He, K. Zhao, X. Chu, Automl: A survey of the state-of-the-art, Knowledge-Based Systems 212 (2021) 106622.

[180] J. Waring, C. Lindvall, R. Umeton, Automated machine learning: Review of the state-of-the-art and opportunities for healthcare, Artificial Intelligence in Medicine 104 (2020) 101822.

[181] A. Mnih, R. R. Salakhutdinov, Probabilistic matrix factorization, Advances in neural information processing systems 20 (2007) 1257–1264.

[182] M. E. Tipping, C. M. Bishop, Probabilistic principal component analysis, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 61 (1999) 611–622.

[183] R. Salakhutdinov, A. Mnih, Bayesian probabilistic matrix factorization using markov chain monte carlo, in: Proceedings of the 25th international conference on Machine learning, 2008, pp. 880–887.

[184] S. Z. Li, Markov random field models in computer vision, in: European conference on computer vision, Springer, 1994, pp. 361–370.

[185] F. Peng, J. Lu, Y. Wang, R. Yi-Da Xu, C. Ma, J. Yang, N-dimensional markov random field prior for cold-start recommendation, Neurocomputing 191 (2016) 187–199.

[186] V. E. Johnson, J. H. Albert, Ordinal data modeling, Springer Science & Business Media, 2006.

[187] P. Gopalan, L. Charlin, D. M. Blei, et al., Content-based recommendations with poisson factorization., in: NIPS, volume 14, 2014, pp. 3176–3184.

[188] A. Acharya, J. Ghosh, M. Zhou, Nonparametric bayesian factor analysis for dynamic count matrices, in: Artificial Intelligence and Statistics, PMLR, 2015, pp. 1–9.

[189] Q. Yao, M. Wang, Y. Chen, W. Dai, Y.-F. Li, W.-W. Tu, Q. Yang, Y. Yu, Taking human out of learning applications: A survey on automated machine learning, arXiv preprint arXiv:1810.13306 (2018).

[190] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, arXiv preprint arXiv:1205.2618 (2012).

[191] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted boltzmann machines for collaborative filtering, in: Proceedings of the 24th international conference on Machine learning, 2007, pp. 791–798.

[192] S.-C. Wang, Artificial neural network, in: Interdisciplinary computing in java programming, Springer, 2003, pp. 81–100.

[193] P. Covington, J. Adams, E. Sargin, Deep neural networks for youtube recommendations, in: Proceedings of the 10th ACM conference on recommender systems, 2016, pp. 191–198.

[194] Y. Takefuji, Neural network parallel computing, volume 164, Springer Science & Business Media, 2012.

[195] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th international conference on world wide web, 2017, pp. 173–182.

[196] D. Kim, C. Park, J. Oh, S. Lee, H. Yu, Convolutional matrix factorization for document context-aware recommendation, in: Proceedings of the 10th ACM conference on recommender systems, 2016, pp. 233–240.

[197] Y. Li, Z. Hao, H. Lei, Survey of convolutional neural network, Journal of Computer Applications 36 (2016) 2508–2515.

[198] G. Pandey, A. Dukkipati, To go deep or wide in learning?, in: Artificial Intelligence and Statistics, PMLR, 2014, pp. 724–732.

[199] H.-J. Xue, X. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems., in: IJCAI, volume 17, Melbourne, Australia, 2017, pp. 3203–3209.

[200] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, Deepfm: a factorization-machine based neural network for ctr prediction, arXiv preprint arXiv:1703.04247 (2017).

[201] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, G. Sun, xdeepfm: Combining explicit and implicit feature interactions for recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1754–1763.

[202] R. Salakhutdinov, G. Hinton, Deep boltzmann machines, in: Artificial intelligence and statistics, PMLR, 2009, pp. 448–455.

[203] G. Hu, H. Li, Y. Xia, L. Luo, A deep boltzmann machine and multi-grained scanning forest ensemble collaborative method and its application to industrial fault diagnosis, Computers in Industry 100 (2018) 287–296.

[204] D. Liang, L. Charlin, J. McInerney, D. M. Blei, Modeling user exposure in recommendation, in: Proceedings of the 25th international conference on World Wide Web, 2016, pp. 951–961.

[205] R. Devooght, N. Kourtellis, A. Mantrach, Dynamic matrix factorization with priors on unknown values, in: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp. 189–198.

[206] B. Marlin, R. S. Zemel, S. Roweis, M. Slaney, Collaborative filtering and the missing at random assumption, arXiv preprint arXiv:1206.5267 (2012).

[207] H. Steck, Training and testing of recommender systems on data missing not at random, in: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 713–722.

[208] T. Chen, Y. Sun, Y. Shi, L. Hong, On sampling strategies for neural network-based collaborative filtering, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 767–776.

[209] C. Elkan, K. Noto, Learning classifiers from only positive and unlabeled data, in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008, pp. 213–220.

[210] D. Jannach, L. Lerche, M. Zanker, Recommending based on implicit feedback, in: Social Information Access, Springer, 2018, pp. 510–569.

[211] Y. Saito, S. Yaginuma, Y. Nishino, H. Sakata, K. Nakata, Unbiased recommender learning from missing-not-at-random implicit feedback, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 501–509.

[212] X. He, H. Zhang, M.-Y. Kan, T.-S. Chua, Fast matrix factorization for online recommendation with implicit feedback, in: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, pp. 549–558.

[213] T. Man, H. Shen, X. Jin, X. Cheng, Cross-domain recommendation: An embedding and mapping approach., in: IJCAI, 2017, pp. 2464–2470.

[214] T. Zang, Y. Zhu, H. Liu, R. Zhang, J. Yu, A survey on cross-domain recommendation: Taxonomies, methods, and future directions, arXiv preprint arXiv:2108.03357 (2021).

[215] F. Zhu, Y. Wang, C. Chen, J. Zhou, L. Li, G. Liu, Cross-domain recommendation: challenges, progress, and prospects, arXiv preprint arXiv:2103.01696 (2021).

[216] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on knowledge and data engineering 22 (2009) 1345–1359.

[217] J. Liu, C. Wang, J. Gao, J. Han, Multi-view clustering via joint nonnegative matrix factorization, in: Proceedings of the 2013 SIAM International Conference on Data Mining, SIAM, 2013, pp. 252–260.

[218] M. Long, J. Wang, G. Ding, J. Sun, P. S. Yu, Transfer feature learning with joint distribution adaptation, in: Proceedings of the IEEE international conference on computer vision, 2013, pp. 2200–2207.

[219] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, M. Guo, Multi-task feature learning for knowledge graph enhanced recommendation, in: The World Wide Web Conference, 2019, pp. 2000–2010.

[220] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, A comprehensive survey on transfer learning, Proceedings of the IEEE 109 (2020) 43–76.

[221] P. Li, A. Tuzhilin, Ddtcdr: Deep dual transfer cross domain recommendation, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 331–339.

[222] M. Long, J. Wang, G. Ding, W. Cheng, X. Zhang, W. Wang, Dual transfer learning, in: Proceedings of the 2012 SIAM International Conference on Data Mining, SIAM, 2012, pp. 540–551.

[223] Y. Xia, D. He, T. Qin, L. Wang, N. Yu, T.-Y. Liu, W.-Y. Ma, Dual learning for machine translation, arXiv preprint

arXiv:1611.00179 (2016).

[224] Y. Xia, T. Qin, W. Chen, J. Bian, N. Yu, T.-Y. Liu, Dual supervised learning, in: International Conference on Machine Learning, PMLR, 2017, pp. 3789–3798.

[225] H. Steck, M. Dimakopoulou, N. Riabov, T. Jebara, Admm slim: Sparse recommendations for many users, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 555–563.

[226] Y. Yuan, X. Luo, M. Shang, D. Wu, A generalized and fast-converging non-negative latent factor model for predicting user preferences in recommender systems, in: Proceedings of The Web Conference 2020, 2020, pp. 498–507.

[227] S. Abbar, S. Amer-Yahia, P. Indyk, S. Mahabadi, Real-time recommendation of diverse related articles, in: Proceedings of the 22nd international conference on World Wide Web, 2013, pp. 1–12.

[228] R. Albalawi, T. H. Yeap, M. Benyoucef, Toward a real-time social recommendation system, in: Proceedings of the 11th International Conference on Management of Digital EcoSystems, 2019, pp. 336–340.

[229] Y. Ma, B. Narayanaswamy, H. Lin, H. Ding, Temporal-contextual recommendation in real-time, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 2291–2299.

[230] B. M. Khan, A. Mansha, F. H. Khan, S. Bashir, Collaborative filtering based online recommendation systems: A survey, in: 2017 International Conference on Information and Communication Technologies (ICICT), IEEE, 2017, pp. 125–130.

[231] S. C. Hoi, D. Sahoo, J. Lu, P. Zhao, Online learning: A comprehensive survey, arXiv preprint arXiv:1802.02871 (2018).

[232] G. Ling, H. Yang, I. King, M. R. Lyu, Online learning for collaborative filtering, in: The 2012 International Joint Conference on Neural Networks (IJCNN), IEEE, 2012, pp. 1–8.

[233] Y. Koren, Collaborative filtering with temporal dynamics, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009, pp. 447–456.

[234] A. Vidmer, M. Medo, The essential role of time in network-based recommendation, EPL (Europhysics Letters) 116 (2016) 30007.

[235] W. Yong, Y. Ting, A distance decay and score trends improved collaborative recommendation algorithm, Journal of Guangdong University of Technology (2015) 02.

[236] P. Richtárik, M. Takáč, Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function, Mathematical Programming 144 (2014) 1–38.

[237] S. Boyd, S. P. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.

[238] S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Factorizing personalized markov chains for next-basket recommendation, in: Proceedings of the 19th international conference on World wide web, 2010, pp. 811–820.

[239] H. Zhao, Q. Liu, H. Zhu, Y. Ge, E. Chen, Y. Zhu, J. Du, A sequential approach to market state modeling and analysis in online p2p lending, IEEE Transactions on Systems, Man, and Cybernetics: Systems 48 (2017) 21–33.

[240] R. He, J. McAuley, Fusing similarity models with markov chains for sparse sequential recommendation, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 191–200.

[241] Y.-D. Kim, S. Choi, Nonnegative tucker decomposition, in: 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–8.

[242] X. Wu, Q. Liu, E. Chen, L. He, J. Lv, C. Cao, G. Hu, Personalized next-song recommendation in online karaokes, in: Proceedings of the 7th ACM Conference on Recommender Systems, 2013, pp. 137–140.

[243] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, X. Cheng, Learning hierarchical representation model for nextbasket recommendation, in: Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval, 2015, pp. 403–412.

[244] R. Zhang, Y. Mao, Movie recommendation via markovian factorization of matrix processes, IEEE Access 7 (2019) 13189–13199.

[245] M. Nickel, V. Tresp, H.-P. Kriegel, A three-way model for collective learning on multi-relational data, in: Icml, 2011.

[246] A. P. Singh, G. J. Gordon, Relational learning via collective matrix factorization, in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008, pp. 650–658.

[247] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering., in: Nips, volume 14, 2001, pp. 585–591.

[248] J. B. Tenenbaum, V. De Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, science 290 (2000) 2319–2323.

[249] M. A. Cox, T. F. Cox, Multidimensional scaling, in: Handbook of data visualization, Springer, 2008, pp. 315–347.

[250] R. Jenatton, N. Le Roux, A. Bordes, G. Obozinski, A latent factor model for highly multi-relational data, in: Advances in Neural Information Processing Systems 25 (NIPS 2012), 2012, pp. 3176–3184.

[251] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Neural Information Processing Systems (NIPS), 2013, pp. 1–9.

[252] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 28, 2014.

[253] G. Ji, S. He, L. Xu, K. Liu, J. Zhao, 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), 2015, pp. 687–696.

[254] D. M. Blei, T. L. Griffiths, M. I. Jordan, The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies, Journal of the ACM (JACM) 57 (2010) 1–30.

[255] H. Xiao, M. Huang, Y. Hao, X. Zhu, Transg: A generative mixture model for knowledge graph embedding, arXiv preprint arXiv:1509.05488 (2015).

[256] X. Lin, Y. Liang, F. Giunchiglia, X. Feng, R. Guan, Relation path embedding in knowledge graphs, Neural Computing and Applications 31 (2019) 5629–5639.

[257] U. Naseem, I. Razzak, S. K. Khan, M. Prasad, A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models, Transactions on Asian and Low-Resource Language Information Processing 20 (2021) 1–35.

[258] Y. Bengio, R. Ducharme, P. Vincent, C. Janvin, A neural probabilistic language model, The journal of machine learning research 3 (2003) 1137–1155.

[259] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).

[260] F. Morin, Y. Bengio, Hierarchical probabilistic neural network language model., in: Aistats, volume 5, Citeseer, 2005, pp. 246–252.

[261] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, arXiv preprint arXiv:1310.4546 (2013).

[262] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[263] C. Yang, Z. Liu, Comprehend deepwalk as matrix factorization, arXiv preprint arXiv:1501.00358 (2015).

[264] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th international conference on world wide web, 2015, pp. 1067–1077.

[265] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, J. Tang, Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec, in: Proceedings of the eleventh ACM international conference on web search and data mining, 2018, pp. 459–467.

[266] J. Tang, M. Qu, Q. Mei, Pte: Predictive text embedding through large-scale heterogeneous text networks, in: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp. 1165–1174.

[267] M. Gao, L. Chen, X. He, A. Zhou, Bine: Bipartite network embedding, in: The 41st international ACM SIGIR conference on research & development in information retrieval, 2018, pp. 715–724.

[268] R. Hussein, D. Yang, P. Cudré-Mauroux, Are meta-paths necessary? revisiting heterogeneous graph embeddings, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 437–446.

[269] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, P. S. Yu, A survey on heterogeneous graph embedding: methods, techniques, applications and sources, arXiv preprint arXiv:2011.14867 (2020).

[270] T.-y. Fu, W.-C. Lee, Z. Lei, Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1797–1806.

[271] Y. Dong, N. V. Chawla, A. Swami, metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 135–144.

[272] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, T. S. Huang, Heterogeneous network embedding via deep architectures, in: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp. 119–128.

[273] H. Gui, J. Liu, F. Tao, M. Jiang, B. Norick, J. Han, Large-scale embedding learning in heterogeneous event data, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 907–912.

[274] J. Shang, M. Qu, J. Liu, L. M. Kaplan, J. Han, J. Peng, Meta-path guided embedding for similarity search in large-scale heterogeneous information networks, arXiv preprint arXiv:1610.09769 (2016).

[275] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[276] A. Rakhlin, Convolutional neural networks for sentence classification, GitHub (2016).

[277] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning internal representations by error propagation, Technical Report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[278] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, L. Bottou, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion., Journal of machine learning research 11 (2010).

[279] R. Salakhutdinov, G. Hinton, Semantic hashing, International Journal of Approximate Reasoning 50 (2009) 969–978.

[280] J. Masci, U. Meier, D. Cireşan, J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, in: International conference on artificial neural networks, Springer, 2011, pp. 52–59.

[281] W. Wang, Y. Huang, Y. Wang, L. Wang, Generalized autoencoder: A neural network framework for dimensionality reduction, in: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2014, pp. 490–497.

[282] D. J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in: International conference on machine learning, PMLR, 2014, pp. 1278–1286.

[283] A. Vahdat, J. Kautz, Nvae: A deep hierarchical variational autoencoder, arXiv preprint arXiv:2007.03898 (2020).

[284] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 1225–1234.

[285] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, arXiv preprint arXiv:1409.3215 (2014).

[286] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473 (2014).

[287] K. Cho, A. Courville, Y. Bengio, Describing multimedia content using attention-based encoder-decoder networks, IEEE Transactions on Multimedia 17 (2015) 1875–1886.

[288] J. B. Lee, R. A. Rossi, S. Kim, N. K. Ahmed, E. Koh, Attention models in graphs: A survey, ACM Transactions on Knowledge Discovery from Data (TKDD) 13 (2019) 1–25.

[289] D. Hu, An introductory survey on attention mechanisms in nlp problems, in: Proceedings of SAI Intelligent Systems Conference, Springer, 2019, pp. 432–448.

[290] Y. Kim, C. Denton, L. Hoang, A. M. Rush, Structured attention networks, arXiv preprint arXiv:1702.00887 (2017).

[291] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, arXiv preprint arXiv:1706.03762 (2017).

[292] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).

[293] R. He, W.-C. Kang, J. McAuley, Translation-based recommendation, in: Proceedings of the eleventh ACM conference on recommender systems, 2017, pp. 161–169.

[294] H. Li, Y. Liu, N. Mamoulis, D. S. Rosenblum, Translation-based sequential recommendation for complex users on sparse data, IEEE Transactions on Knowledge and Data Engineering 32 (2019) 1639–1651.

[295] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, J. Han, Personalized entity recommendation: A heterogeneous information network approach, in: Proceedings of the 7th ACM international conference on Web search and data mining, 2014, pp. 283–292.

[296] H. Zhao, Q. Yao, J. Li, Y. Song, D. L. Lee, Meta-graph based recommendation fusion over heterogeneous information networks, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 635–644.

[297] C. Shi, B. Hu, W. X. Zhao, S. Y. Philip, Heterogeneous information network embedding for recommendation, IEEE Transactions on Knowledge and Data Engineering 31 (2018) 357–370.

[298] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, M. Guo, Ripplenet: Propagating user preferences on the knowledge graph for recommender systems, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 417–426.

[299] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, T.-S. Chua, Explainable reasoning over knowledge graphs for recommendation, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019, pp. 5329–5336.

[300] H. Chen, Y. Li, X. Sun, G. Xu, H. Yin, Temporal meta-path guided explainable recommendation, arXiv preprint arXiv:2101.01433 (2021).

[301] S. Sedhain, A. K. Menon, S. Sanner, L. Xie, Autorec: Autoencoders meet collaborative filtering, in: Proceedings of the 24th international conference on World Wide Web, 2015, pp. 111–112.

[302] Y. Wu, C. DuBois, A. X. Zheng, M. Ester, Collaborative denoising auto-encoders for top-n recommender systems, in: Proceedings of the ninth ACM international conference on web search and data mining, 2016, pp. 153–162.

[303] D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: Pro-

ceedings of the 2018 world wide web conference, 2018, pp. 689–698.

[304] X. Li, J. She, Collaborative variational autoencoder for recommender systems, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 305–314.

[305] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, T.-S. Chua, Attentional factorization machines: Learning the weight of feature interactions via attention networks, arXiv preprint arXiv:1708.04617 (2017).

[306] C. Xu, J. Feng, P. Zhao, F. Zhuang, D. Wang, Y. Liu, V. S. Sheng, Long-and short-term self-attention network for sequential recommendation, Neurocomputing 423 (2021) 580–589.

[307] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, D. L. Lee, Billion-scale commodity embedding for e-commerce recommendation in alibaba, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 839–848.

[308] S. Wang, L. Cao, Y. Wang, Q. Z. Sheng, M. Orgun, D. Lian, A survey on session-based recommender systems, arXiv preprint arXiv:1902.04864 (2019).

[309] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, Q. Liu, Shine: Signed heterogeneous information network embedding for sentiment link prediction, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 592–600.

[310] J. He, W. W. Chu, A social network-based recommender system (snrs), in: Data mining for social network data, Springer, 2010, pp. 47–74.

[311] H. Zhang, G. Liu, J. Wu, Social collaborative filtering ensemble, in: Pacific Rim International Conference on Artificial Intelligence, Springer, 2018, pp. 1005–1017.

[312] L. Chen, H. Zhang, J. Wu, Integrating dual user network embedding with matrix factorization for social recommender systems, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.

[313] Z. Ding, X. Li, C. Jiang, M. Zhou, Objectives and state-of-the-art of location-based social network recommender systems, ACM Computing Surveys (CSUR) 51 (2018) 1–28.

[314] H. Werneck, N. Silva, M. C. Viana, F. Mourão, A. C. Pereira, L. Rocha, A survey on point-of-interest recommendation in location-based social networks, in: Proceedings of the Brazilian Symposium on Multimedia and the Web, 2020, pp. 185–192.

[315] W. X. Zhao, S. Li, Y. He, E. Y. Chang, J.-R. Wen, X. Li, Connecting social media to e-commerce: Cold-start product recommendation using microblogging information, IEEE Transactions on Knowledge and Data Engineering 28 (2015) 1147–1159.

[316] C. Tilly, Identities, boundaries and social ties, Routledge, 2015.

[317] M. S. Granovetter, The strength of weak ties, American journal of sociology 78 (1973) 1360–1380.

[318] X. Wang, W. Lu, M. Ester, C. Wang, C. Chen, Social recommendation with strong and weak ties, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 5–14.

[319] H. Wang, M. Terrovitis, N. Mamoulis, Location recommendation in location-based social networks using user check-in data, in: Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems, 2013, pp. 374–383.

[320] A. Ladle, P. Galpern, P. Doyle-Baker, Measuring the use of green space with urban resource selection functions: An application using smartphone gps locations, Landscape and Urban Planning 179 (2018) 107–115.

[321] M. W. Traunmueller, N. Johnson, A. Malik, C. E. Kontokosta, Digital footprints: Using wifi probe and locational data to analyze human mobility trajectories in cities, Computers, Environment and Urban Systems 72 (2018) 4–12.

[322] K. Wang, W. Meng, J. Bian, S. Li, S. Yang, Spatial context-aware user mention behavior modeling for mentionee recommendation, Neural Networks 106 (2018) 152–167.

[323] G. Zhao, P. Lou, X. Qian, X. Hou, Personalized location recommendation by fusing sentimental and spatial context, Knowledge-Based Systems 196 (2020) 105849.

[324] P. Berkhin, Bookmark-coloring algorithm for personalized pagerank computing, Internet Mathematics 3 (2006) 41–62.

[325] G. Ference, M. Ye, W.-C. Lee, Location recommendation for out-of-town users in location-based social networks, in: Proceedings of the 22nd ACM international conference on Information & Knowledge Management, 2013, pp. 721–726.

[326] H. Yin, Y. Sun, B. Cui, Z. Hu, L. Chen, Lcars: a location-content-aware recommender system, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013, pp. 221–229.

[327] Y. Shi, L. Zhu, W. Li, K. Guo, Y. Zheng, Survey on classic and latest textual sentiment analysis articles and techniques, International Journal of Information Technology & Decision Making 18 (2019) 1243–1287.

[328] D. A. Pereira, A survey of sentiment analysis in the portuguese language, Artificial Intelligence Review 54 (2021) 1087–1115.

[329] Z. Yao, Exploiting human mobility patterns for point-of-interest recommendation, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 757–758.

[330] S. Jiang, X. Qian, J. Shen, Y. Fu, T. Mei, Author topic model-based collaborative filtering for personalized poi recommendations, IEEE transactions on multimedia 17 (2015) 907–918.

[331] S. Auer, V. Kovtun, M. Prinz, A. Kasprzik, M. Stocker, M. E. Vidal, Towards a knowledge graph for science, in: Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, 2018, pp. 1–6.

[332] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, A comprehensive survey on graph neural networks, IEEE transactions on neural networks and learning systems (2020).

[333] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81.

[334] F. Wang, C. Zhang, Label propagation through linear neighborhoods, IEEE Transactions on Knowledge and Data Engineering 20 (2007) 55–67.

[335] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: The world wide web conference, 2019, pp. 3307–3313.

[336] T. Joachims, et al., Transductive inference for text classification using support vector machines, in: Icml, volume 99, 1999, pp. 200–209.

[337] M. Belkin, P. Niyogi, V. Sindhwani, On manifold regularization., in: AISTATS, volume 1, 2005.

[338] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, arXiv preprint arXiv:1706.02216 (2017).

[339] D. Cheng, Y. Cheng, Y. Liu, R. Peng, S.-H. Teng, Efficient sampling for gaussian graphical models via spectral sparsification, in: Conference on Learning Theory, PMLR, 2015, pp. 364–390.

[340] J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, J. Tang, Netsmf: Large-scale network embedding as sparse matrix factorization, in: The World Wide Web Conference, 2019, pp. 1509–1520.

[341] H. Liu, Y. Wu, Y. Yang, Analogical inference for multi-relational embeddings, in: International conference on machine learning, PMLR, 2017, pp. 2168–2178.

[342] M. Nickel, L. Rosasco, T. Poggio, Holographic embeddings of knowledge graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 30, 2016.

[343] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, A survey of convolutional neural networks: analysis, applications, and prospects, IEEE Transactions on Neural Networks and Learning Systems (2021).

[344] C. Xu, D. Tao, C. Xu, A survey on multi-view learning, arXiv preprint arXiv:1304.5634 (2013).

[345] Y. Shi, F. Han, X. He, X. He, C. Yang, J. Luo, J. Han, mvn2vec: Preservation and collaboration in multi-view network embedding, arXiv preprint arXiv:1801.06597 (2018).

[346] K. Chaudhuri, S. M. Kakade, K. Livescu, K. Sridharan, Multi-view clustering via canonical correlation analysis, in: Proceedings of the 26th annual international conference on machine learning, 2009, pp. 129–136.

[347] A. Kumar, P. Rai, H. Daume, Co-regularized multi-view spectral clustering, Advances in neural information processing systems 24 (2011) 1413–1421.

[348] T. Xia, D. Tao, T. Mei, Y. Zhang, Multiview spectral embedding, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 40 (2010) 1438–1446.

[349] D. Greene, P. Cunningham, A matrix factorization approach for integrating multiple data views, in: Joint European conference on machine learning and knowledge discovery in databases, Springer, 2009, pp. 423–438.

[350] H. Zhang, L. Qiu, L. Yi, Y. Song, Scalable multiplex network embedding., in: IJCAI, volume 18, 2018, pp. 3082–3088.

[351] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, J. Han, An attention-based collaboration framework for multi-view network representation learning, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1767–1776.

[352] A. R. Benson, D. F. Gleich, J. Leskovec, Higher-order organization of complex networks, Science 353 (2016) 163–166.

[353] M. De Domenico, V. Nicosia, A. Arenas, V. Latora, Structural reducibility of multilayer networks, Nature communications 6 (2015) 1–9.

[354] Y. Hu, S. Havlin, H. A. Makse, Conditions for viral influence spreading through multiplex correlated social networks, Physical Review X 4 (2014) 021031.

[355] B. Perozzi, V. Kulkarni, H. Chen, S. Skiena, Don't walk, skip! online learning of multi-scale network embeddings, in: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, 2017, pp. 258–265.

[356] L. F. Ribeiro, P. H. Saverese, D. R. Figueiredo, struc2vec: Learning node representations from structural identity, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 385–394.

[357] M. Ouyang, Review on modeling and simulation of interdependent critical infrastructure systems, Reliability engineering & System safety 121 (2014) 43–60.

[358] S. Nambisan, Platforms for collaboration, Stanford Social Innovation Review 7 (2009) 44–49.

[359] L. Xu, X. Wei, J. Cao, P. S. Yu, Embedding of embedding (eoe) joint embedding for coupled heterogeneous networks, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017, pp. 741–749.

[360] J. Li, C. Chen, H. Tong, H. Liu, Multi-layered network embedding, in: Proceedings of the 2018 SIAM International Conference on Data Mining, SIAM, 2018, pp. 684–692.

[361] W. Liu, P.-Y. Chen, S. Yeung, T. Suzumura, L. Chen, Principled multilayer network embedding, in: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), IEEE, 2017, pp. 134–141.

[362] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, J. Tang, Representation learning for attributed multiplex heterogeneous network, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1358–1368.

[363] J. Ni, S. Chang, X. Liu, W. Cheng, H. Chen, D. Xu, X. Zhang, Co-regularized deep multi-network embedding, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 469–478.

[364] X. Li, N. Du, H. Li, K. Li, J. Gao, A. Zhang, A deep learning approach to link prediction in dynamic networks, in: Proceedings of the 2014 SIAM International Conference on Data Mining, SIAM, 2014, pp. 289–297.

[365] Y. Lu, X. Wang, C. Shi, P. S. Yu, Y. Ye, Temporal network embedding with micro-and macro-dynamics, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 469–478.

[366] Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, J. Wu, Embedding temporal network via neighborhood formation, in: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 2857–2866.

[367] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, K. Achan, Inductive representation learning on temporal graphs, arXiv preprint arXiv:2002.07962 (2020).

[368] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, P. Poupart, Representation learning for dynamic graphs: A survey., Journal of Machine Learning Research 21 (2020) 1–73.

[369] V. W. Anelli, T. Di Noia, E. Di Sciascio, A. Ragone, J. Trotta, How to make latent factors interpretable by feeding factorization machines with knowledge graphs, in: International Semantic Web Conference, Springer, 2019, pp. 38–56.

[370] F. Zhang, N. J. Yuan, D. Lian, X. Xie, W.-Y. Ma, Collaborative knowledge base embedding for recommender systems, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 353–362.

[371] X. Wang, X. He, Y. Cao, M. Liu, T.-S. Chua, Kgat: Knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 950–958.

[372] C. Ma, L. Ma, Y. Zhang, H. Wu, X. Liu, M. Coates, Knowledge-enhanced top-k recommendation in poincar\'e ball, arXiv preprint arXiv:2101.04852 (2021).

[373] R. Togashi, M. Otani, S. Satoh, Alleviating cold-start problems in recommendation through pseudo-labelling over knowledge graph, arXiv preprint arXiv:2011.05061 (2020).

[374] D.-H. Lee, et al., Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks, in: Workshop on challenges in representation learning, ICML, volume 3, 2013.

[375] X. Wang, Y. Xu, X. He, Y. Cao, M. Wang, T.-S. Chua, Reinforced negative sampling over knowledge graph for recommendation, in: Proceedings of The Web Conference 2020, 2020, pp. 99–109.

[376] C. Chen, M. Zhang, W. Ma, Y. Liu, S. Ma, Jointly non-sampling learning for knowledge graph enhanced recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 189–198.

[377] Y. Xian, Z. Fu, S. Muthukrishnan, G. De Melo, Y. Zhang, Reinforcement knowledge graph reasoning for explainable recommendation, in: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, 2019, pp. 285–294.

[378] W. Lei, X. He, M. de Rijke, T.-S. Chua, Conversational recommendation: Formulation, methods, and evaluation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 2425–2428.

[379] D. Jannach, A. Manzoor, W. Cai, L. Chen, A survey on conversational recommender systems, ACM Computing Surveys (CSUR) 54 (2021) 1–36.

[380] C. Gao, W. Lei, X. He, M. de Rijke, T.-S. Chua, Advances and challenges in conversational recommender systems: A survey, arXiv preprint arXiv:2101.09459 (2021).

[381] K. Zhou, Y. Zhou, W. X. Zhao, X. Wang, J.-R. Wen, Towards topic-guided conversational recommender system, arXiv preprint arXiv:2010.04125 (2020).

[382] J. Li, M. Galley, C. Brockett, J. Gao, B. Dolan, A diversity-promoting objective function for neural conversation models, arXiv preprint arXiv:1510.03055 (2015).

[383] O. Vinyals, Q. Le, A neural conversational model, arXiv preprint arXiv:1506.05869 (2015).

[384] Y. Zhang, X. Chen, Q. Ai, L. Yang, W. B. Croft, Towards conversational search and recommendation: System ask, user respond, in: Proceedings of the 27th acm international conference on information and knowledge management, 2018, pp. 177–186.

[385] Q. Chen, J. Lin, Y. Zhang, M. Ding, Y. Cen, H. Yang, J. Tang, Towards knowledge-based recommender dialog system, arXiv preprint arXiv:1908.05391 (2019).

[386] W. Lei, X. He, Y. Miao, Q. Wu, R. Hong, M.-Y. Kan, T.-S. Chua, Estimation-action-reflection: Towards deep interaction between conversational and recommender systems, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 304–312.

[387] X. Zhang, H. Xie, H. Li, J. Lui, Toward building conversational recommender systems: A contextual bandit approach, arXiv preprint arXiv:1906.01219 (2019).

[388] J. He, H. H. Zhuo, J. Law, Distributed-representation based hybrid recommender system with short item descriptions, arXiv preprint arXiv:1703.04854 (2017).

[389] R. Li, S. Kahou, H. Schulz, V. Michalski, L. Charlin, C. Pal, Towards deep conversational recommendations, arXiv preprint arXiv:1812.07617 (2018).

[390] K. Zhou, W. X. Zhao, S. Bian, Y. Zhou, J.-R. Wen, J. Yu, Improving conversational recommender systems via knowledge graph based semantic fusion, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1006–1014.

[391] J. Tang, T. Zhao, C. Xiong, X. Liang, E. P. Xing, Z. Hu, Target-guided open-domain conversation, arXiv preprint arXiv:1905.11553 (2019).

[392] D. Kang, A. Balakrishnan, P. Shah, P. Crook, Y.-L. Boureau, J. Weston, Recommendation as a communication game: Self-supervised bot-play for goal-oriented dialogue, arXiv preprint arXiv:1909.03922 (2019).

[393] Z. Liu, H. Wang, Z.-Y. Niu, H. Wu, W. Che, T. Liu, Towards conversational recommendation over multi-type dialogs, arXiv preprint arXiv:2005.03954 (2020).

[394] M. Karimi, D. Jannach, M. Jugovac, News recommender systems–survey and roads ahead, Information Processing & Management 54 (2018) 1203–1227.

[395] M. Li, L. Wang, A survey on personalized news recommendation technology, IEEE Access 7 (2019) 145861–145879.

[396] H. Abdollahpouri, E. C. Malthouse, J. A. Konstan, B. Mobasher, J. Gilbert, Toward the next generation of news recommender systems, in: Companion Proceedings of the Web Conference 2021, 2021, pp. 402–406.

[397] S. Raza, C. Ding, News recommender system: a review of recent progress, challenges, and opportunities, Artificial Intelligence Review (2021) 1–52.

[398] H. Wang, F. Zhang, X. Xie, M. Guo, Dkn: Deep knowledge-aware network for news recommendation, in: Proceedings of the 2018 world wide web conference, 2018, pp. 1835–1844.

[399] S. Albawi, T. A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, in: 2017 International Conference on Engineering and Technology (ICET), Ieee, 2017, pp. 1–6.

[400] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, K. Li, Y. Chen, Y. Lu, H. Wang, C. Tian, X. Pan, et al., Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms, arXiv preprint arXiv:2011.01731 (2020).

[401] N. Hug, Surprise: A python library for recommender systems, Journal of Open Source Software 5 (2020) 2174.

[402] A. Gunawardana, G. Shani, A survey of accuracy evaluation metrics of recommendation tasks., Journal of Machine Learning Research 10 (2009).

[403] W. Krichene, S. Rendle, On sampled metrics for item recommendation, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1748–1757.

[404] C. J. Willmott, K. Matsuura, Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance, Climate research 30 (2005) 79–82.

[405] Y. Wang, L. Wang, Y. Li, D. He, W. Chen, T.-Y. Liu, A theoretical analysis of ndcg ranking measures, in: Proceedings of the 26th annual conference on learning theory (COLT 2013), volume 8, 2013, p. 6.

[406] Z.-D. Zhao, M.-S. Shang, User-based collaborative-filtering recommendation algorithms on hadoop, in: 2010 third international conference on knowledge discovery and data mining, IEEE, 2010, pp. 478–481.

[407] N. Srivastava, R. Salakhutdinov, et al., Multimodal learning with deep boltzmann machines., in: NIPS, volume 1, Citeseer, 2012, p. 2.

[408] J. Wang, A. P. De Vries, M. J. Reinders, Unifying user-based and item-based collaborative filtering approaches by similarity fusion, in: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, 2006, pp. 501–508.

[409] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, Y.-C. Zhang, Solving the apparent diversity-accuracy

dilemma of recommender systems, Proceedings of the National Academy of Sciences 107 (2010) 4511–4515.

[410] Y. Zhou, L. Lü, W. Liu, J. Zhang, The power of ground user in recommender systems, PLoS One 8 (2013) e70094.

[411] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International conference on machine learning, PMLR, 2013, pp. 1310–1318.

[412] Y. Wang, Q. Yao, J. T. Kwok, L. M. Ni, Generalizing from a few examples: A survey on few-shot learning, ACM Computing Surveys (CSUR) 53 (2020) 1–34.

[413] N. Bendre, H. T. Marín, P. Najafirad, Learning from few samples: A survey, arXiv preprint arXiv:2007.15484 (2020).

[414] P. De Handschutter, N. Gillis, X. Siebert, A survey on deep matrix factorizations, Computer Science Review 42 (2021) 100423.

[415] A. E. Sizemore, C. Giusti, A. Kahn, J. M. Vettel, R. F. Betzel, D. S. Bassett, Cliques and cavities in the human connectome, Journal of computational neuroscience 44 (2018) 115–145.

[416] D. Shi, L. Lü, G. Chen, Totally homogeneous networks, National Science Review 6 (2019) 962–969.

[417] F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G. Young, G. Petri, Networks beyond pairwise interactions: structure and dynamics, Physics Reports 874 (2020) 1–92.

[418] D. Shi, Z. Chen, X. Sun, Q. Chen, Y. Lou, G. Chen, Computing cliques and cavities in networks, arXiv preprint arXiv:2101.00536 (2021).

[419] X. Ouvrard, Hypergraphs: an introduction and review, arXiv preprint arXiv:2002.05014 (2020).

[420] M. Aledhari, R. Razzak, R. M. Parizi, F. Saeed, Federated learning: A survey on enabling technologies, protocols, and applications, IEEE Access 8 (2020) 140699–140725.

[421] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Generation Computer Systems 115 (2021) 619–640.

[422] B. Yi, X. Shen, H. Liu, Z. Zhang, W. Zhang, S. Liu, N. Xiong, Deep matrix factorization with implicit feedback embedding for recommendation system, IEEE Transactions on Industrial Informatics 15 (2019) 4591–4601.

[423] O. Barkan, N. Koenigstein, Item2vec: neural item embedding for collaborative filtering, in: 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, 2016, pp. 1–6.

Appendix A Hyper-parameter settings in experiments

Table A1: Hyper-parameter settings. The evaluated models' hyper-parameter settings in Sec. 6.2 follow the principle of making a balance between model's recommendation accuracy and computing efficiency. As for the conventional recommendation models, on the three experimental data sets the k-nearest neighbors (K) of UserKNN, ItemKNN and HybridKNN are set to 10 and the combination ratio (α) of HybridKNN and HybridS is set to 0.5. As for graph embedding-based recommendation models, the optimal embedding dimension (dim) is searched in the space of $[4, 8, 16, 32, 64, 128]$ and the optimal learning rate (lr) is searched in the space of $[0.1, 0.05, 0.01, 0.005, 0.001]$. In addition, as for deep learning-based models, their layers and dropout rate are set by experience. More details are shown below, where n.a. indicates that a model failed to implement on a data set.

	MovieLens 100K	**MovieLens 1M**	**Jester 617K**
UserKNN	K=10	K=10	K=10
ItemKNN	K=10	K=10	K=10
HybridKNN	K=10, $\alpha = 0.5$	K=10, $\alpha = 0.5$	K=10, $\alpha = 0.5$
HybridS	$\alpha = 0.5$	$\alpha = 0.5$	$\alpha = 0.5$
FunkSVD	dim=4, lr=0.01	dim=8, lr=0.01	n.a.
PMF	dim=16, lr=0.01	dim=8, lr=0.01	dim=64, lr=0.01
FM	dim=8, lr=0.01	dim=4, lr=0.01	dim=4, lr=0.01
AutoRec	dim=4, lr=0.01	dim=32, lr=0.001	dim=8, lr=0.05
GMF	dim=128, lr=0.01	dim=128, lr=0.005	dim=4, lr=0.05
MLP	dim=64, lr=0.05, layers=3, dropout=0.5	dim=32, lr=0.005, layers=3, dropout=0.5	dim=16, lr=0.01, layers=3, dropout=0.5
NCF	dim=128, lr=0.01,0.01,0.005, layers=3, dropout=0.5	dim=64, lr=0.01,0.01,0.005, layers=3, dropout=0.5	dim=128, lr=0.01,0.01,0.01, layers=3, dropout=0.5
TransE	dim=128, lr=0.01	dim=64, lr=0.005	dim=128, lr=0.1

Appendix B Hyper-parameter tuning results in experiments

Table A2: Hyper-parameter tuning results on MovieLens 100K for predicting explicit user-item interactions. The notation n.a. indicates that a model failed to implement on the data set.

Metric	MAE						RMSE					
Model	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128
UserKNN	0.702						0.912					
ItemKNN	0.718						0.936					
FunkSVD	0.709	0.724	0.736	0.750	0.780	0.813	0.926	0.944	0.961	0.974	1.014	1.052
PMF	0.759	0.758	0.743	0.785	0.760	0.825	0.998	0.996	0.972	1.034	0.989	1.077
FM	0.692	0.692	0.697	0.714	0.731	0.759	0.890	0.889	0.896	0.915	0.935	0.973
AutoRec	0.733	0.756	0.737	0.792	0.813	0.787	0.947	0.980	0.955	1.020	1.042	1.009

Table A3: Hyper-parameter tuning results on MovieLens 1M for predicting explicit user-item interactions.

Metric	MAE						RMSE					
Model	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128
UserKNN	0.773						0.978					
ItemKNN	0.718						0.907					
FunkSVD	0.699	0.694	0.700	0.710	0.719	0.731	0.892	0.887	0.894	0.905	0.916	0.930
PMF	0.695	0.682	0.707	0.699	0.710	0.698	0.883	0.869	0.897	0.888	0.901	0.887
FM	0.685	0.685	0.699	0.715	0.736	0.760	0.869	0.873	0.889	0.907	0.929	0.959
AutoRec	0.776	0.736	0.732	0.722	0.727	0.758	0.973	0.927	0.923	0.917	0.923	0.960

Table A4: Hyper-parameter tuning results on Jester 617K for predicting explicit user-item interactions.

Metric	MAE						RMSE					
Model	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128
UserKNN	0.857						1.053					
ItemKNN	0.823						1.015					
FunkSVD	n.a.						n.a.					
PMF	0.828	0.828	0.818	0.819	0.817	0.860	1.041	1.040	1.026	1.027	1.026	1.082
FM	0.813	0.816	0.821	0.827	0.836	0.852	1.008	1.011	1.016	1.022	1.032	1.051
AutoRec	0.925	0.911	0.912	1.123	0.949	0.970	1.119	1.106	1.107	1.400	1.139	1.162

Table A5: Hyper-parameter tuning results on MovieLens 100K for predicting implicit user-item interactions.

Metric		Recall						NDCG					
Model	list length	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128
User-KNN	@10			0.090						0.044			
	@20			0.147						0.059			
	@30			0.188						0.067			
	@40			0.222						0.074			
	@50			0.252						0.079			
Item-KNN	@10			0.092						0.046			
	@20			0.146						0.060			
	@30			0.185						0.068			
	@40			0.220						0.075			
	@50			0.245						0.079			
Hybrid-KNN	@10			0.105						0.053			
	@20			0.168						0.068			
	@30			0.212						0.078			
	@40			0.247						0.084			
	@50			0.276						0.090			
ProbS	@10			0.087						0.043			
	@20			0.139						0.056			
	@30			0.177						0.065			
	@40			0.210						0.071			
	@50			0.236						0.076			
HybridS	@10			0.092						0.046			
	@20			0.149						0.060			
	@30			0.191						0.069			
	@40			0.223						0.075			
	@50			0.254						0.081			
GMF	@10	0.023	0.029	0.037	0.044	0.055	0.060	0.011	0.014	0.018	0.020	0.025	0.029
	@20	0.046	0.056	0.066	0.080	0.095	0.103	0.017	0.021	0.025	0.029	0.035	0.040
	@30	0.067	0.079	0.091	0.109	0.128	0.139	0.021	0.026	0.030	0.036	0.043	0.047
	@40	0.091	0.100	0.116	0.136	0.152	0.172	0.026	0.030	0.035	0.041	0.047	0.054
	@50	0.113	0.121	0.136	0.159	0.176	0.199	0.030	0.034	0.039	0.045	0.051	0.058
MLP	@10	0.002	0.018	0.035	0.030	0.042	0.042	0.001	0.008	0.016	0.015	0.020	0.021
	@20	0.003	0.039	0.066	0.055	0.077	0.078	0.001	0.013	0.024	0.022	0.029	0.030
	@30	0.006	0.057	0.094	0.082	0.104	0.103	0.002	0.017	0.030	0.027	0.035	0.035
	@40	0.009	0.076	0.121	0.104	0.128	0.128	0.002	0.020	0.035	0.032	0.040	0.040
	@50	0.011	0.096	0.143	0.128	0.149	0.148	0.003	0.024	0.039	0.036	0.043	0.043
NCF	@10	0.036	0.031	0.037	0.046	0.057	0.061	0.016	0.014	0.018	0.022	0.027	0.029
	@20	0.066	0.056	0.068	0.081	0.099	0.104	0.024	0.021	0.025	0.031	0.038	0.040
	@30	0.091	0.076	0.093	0.109	0.136	0.141	0.029	0.025	0.031	0.037	0.046	0.048
	@40	0.117	0.093	0.117	0.135	0.164	0.169	0.034	0.028	0.035	0.042	0.051	0.053
	@50	0.139	0.113	0.136	0.158	0.189	0.195	0.038	0.032	0.039	0.046	0.056	0.058
TransE	@10	0.001	0.002	0.009	0.014	0.026	0.034	0.001	0.001	0.004	0.007	0.013	0.016
	@20	0.003	0.003	0.017	0.025	0.044	0.057	0.001	0.001	0.006	0.010	0.018	0.022
	@30	0.005	0.005	0.024	0.036	0.060	0.076	0.002	0.002	0.008	0.012	0.021	0.026
	@40	0.007	0.006	0.030	0.047	0.075	0.094	0.002	0.002	0.009	0.014	0.024	0.029
	@50	0.008	0.007	0.036	0.055	0.089	0.108	0.002	0.002	0.010	0.015	0.026	0.032

Table A6: Hyper-parameter tuning results on MovieLens 1M for predicting implicit user-item interactions. The notation '-' means that a model's requirement for computing resources on the data set was beyond that of the author's experimental devices as a consequence of embedding vector's large dimension.

Metric		Recall						NDCG					
Model	list length	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128
User-KNN	@10	0.095						0.047					
	@20	0.157						0.062					
	@30	0.205						0.073					
	@40	0.246						0.080					
	@50	0.281						0.087					
Item-KNN	@10	0.092						0.046					
	@20	0.154						0.061					
	@30	0.204						0.072					
	@40	0.246						0.080					
	@50	0.283						0.086					
Hybrid-KNN	@10	0.109						0.054					
	@20	0.181						0.072					
	@30	0.236						0.084					
	@40	0.283						0.093					
	@50	0.323						0.100					
ProbS	@10	0.077						0.040					
	@20	0.124						0.051					
	@30	0.164						0.060					
	@40	0.199						0.067					
	@50	0.231						0.072					
HybridS	@10	0.089						0.046					
	@20	0.142						0.059					
	@30	0.187						0.069					
	@40	0.225						0.076					
	@50	0.260						0.083					
GMF	@10	0.075	0.071	0.071	0.074	0.078	0.079	0.037	0.034	0.033	0.035	0.038	0.038
	@20	0.126	0.124	0.124	0.130	0.136	0.138	0.050	0.047	0.047	0.049	0.052	0.052
	@30	0.167	0.168	0.168	0.176	0.183	0.186	0.058	0.057	0.056	0.059	0.062	0.063
	@40	0.203	0.207	0.207	0.217	0.224	0.228	0.065	0.064	0.064	0.067	0.070	0.071
	@50	0.235	0.242	0.242	0.253	0.260	0.265	0.071	0.071	0.070	0.073	0.077	0.078
MLP	@10	0.039	0.060	0.060	0.074	0.024	-	0.018	0.029	0.030	0.036	0.011	-
	@20	0.074	0.107	0.107	0.127	0.046	-	0.027	0.040	0.042	0.049	0.016	-
	@30	0.107	0.146	0.146	0.170	0.066	-	0.034	0.049	0.051	0.058	0.021	-
	@40	0.139	0.181	0.181	0.207	0.087	-	0.040	0.056	0.057	0.066	0.025	-
	@50	0.168	0.213	0.213	0.241	0.106	-	0.045	0.061	0.063	0.072	0.028	-
NCF	@10	0.076	0.080	0.080	0.080	0.083	-	0.037	0.039	0.040	0.039	0.041	-
	@20	0.126	0.135	0.135	0.136	0.141	-	0.050	0.053	0.054	0.053	0.055	-
	@30	0.168	0.181	0.181	0.184	0.188	-	0.059	0.063	0.064	0.063	0.065	-
	@40	0.204	0.222	0.222	0.223	0.228	-	0.066	0.071	0.071	0.071	0.073	-
	@50	0.235	0.257	0.257	0.258	0.264	-	0.071	0.077	0.078	0.077	0.080	-
TransE	@10	0.003	0.003	0.003	0.003	0.010	0.009	0.001	0.001	0.001	0.001	0.005	0.004
	@20	0.006	0.006	0.006	0.006	0.018	0.017	0.002	0.002	0.002	0.002	0.007	0.006
	@30	0.009	0.009	0.009	0.009	0.026	0.024	0.003	0.003	0.003	0.003	0.008	0.008
	@40	0.012	0.012	0.012	0.012	0.034	0.031	0.003	0.003	0.003	0.003	0.010	0.009
	@50	0.015	0.015	0.015	0.015	0.041	0.038	0.004	0.004	0.004	0.004	0.011	0.010

Table A7: Hyper-parameter tuning results on Jester 617K for predicting implicit user-item interactions.

Metric		Recall						NDCG					
Model	list length	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128	dim =4	dim =8	dim =16	dim =32	dim =64	dim =128
User-KNN	@10			0.511						0.279			
	@20			0.648						0.314			
	@30			0.703						0.326			
	@40			0.771						0.339			
	@50			0.842						0.352			
Item-KNN	@10			0.590						0.309			
	@20			0.784						0.358			
	@30			0.843						0.371			
	@40			0.879						0.378			
	@50			0.911						0.384			
Hybrid-KNN	@10			0.591						0.316			
	@20			0.768						0.361			
	@30			0.850						0.378			
	@40			0.886						0.385			
	@50			0.918						0.391			
ProbS	@10			0.628						0.338			
	@20			0.831						0.390			
	@30			0.881						0.401			
	@40			0.916						0.407			
	@50			0.948						0.413			
HybridS	@10			0.641						0.347			
	@20			0.831						0.395			
	@30			0.881						0.406			
	@40			0.917						0.413			
	@50			0.949						0.419			
GMF	@10	0.568	0.567	0.555	0.575	0.578	0.574	0.285	0.275	0.263	0.278	0.276	0.289
	@20	0.816	0.803	0.824	0.823	0.824	0.825	0.349	0.335	0.331	0.342	0.339	0.353
	@30	0.873	0.862	0.871	0.867	0.875	0.872	0.361	0.348	0.342	0.351	0.350	0.363
	@40	0.910	0.902	0.899	0.891	0.906	0.900	0.368	0.356	0.347	0.356	0.356	0.369
	@50	0.943	0.937	0.923	0.914	0.932	0.925	0.374	0.362	0.351	0.360	0.360	0.374
MLP	@10	0.564	0.551	0.571	0.569	0.564	0.527	0.266	0.268	0.287	0.282	0.257	0.259
	@20	0.827	0.828	0.829	0.828	0.778	0.819	0.333	0.340	0.353	0.349	0.312	0.333
	@30	0.880	0.879	0.879	0.879	0.823	0.879	0.344	0.351	0.364	0.360	0.322	0.346
	@40	0.915	0.915	0.915	0.915	0.858	0.915	0.351	0.358	0.371	0.367	0.328	0.353
	@50	0.947	0.948	0.947	0.947	0.889	0.945	0.357	0.363	0.377	0.372	0.334	0.358
NCF	@10	0.570	0.585	0.567	0.581	0.581	0.580	0.269	0.288	0.277	0.275	0.303	0.293
	@20	0.827	0.825	0.828	0.824	0.824	0.827	0.335	0.350	0.344	0.337	0.365	0.357
	@30	0.876	0.876	0.878	0.874	0.872	0.879	0.346	0.360	0.355	0.348	0.375	0.368
	@40	0.909	0.908	0.914	0.905	0.902	0.914	0.352	0.367	0.362	0.354	0.381	0.375
	@50	0.937	0.936	0.945	0.929	0.928	0.947	0.357	0.372	0.368	0.358	0.386	0.380
TransE	@10	0.134	0.122	0.162	0.137	0.166	0.255	0.062	0.056	0.077	0.064	0.078	0.120
	@20	0.254	0.238	0.287	0.257	0.297	0.426	0.092	0.085	0.109	0.094	0.111	0.163
	@30	0.361	0.352	0.402	0.373	0.417	0.546	0.115	0.109	0.133	0.118	0.137	0.189
	@40	0.468	0.465	0.508	0.486	0.528	0.643	0.135	0.131	0.153	0.140	0.158	0.207
	@50	0.575	0.577	0.608	0.593	0.629	0.726	0.155	0.151	0.171	0.160	0.176	0.222

Table A8: The convergence steps of the machine learning-based baselines on the three data sets under specific hyper-parameters as shown in Tab. A1. Among them, since NCF consists of three components, their convergence steps with different embedding dimensions on the three data sets are presented by three rows, respectively. Note that n.a. indicates that a model failed to implement on the data set, and '-' means that a model's requirement for computing resources on the data set was beyond that of the author's experimental devices as a consequence of embedding vector's large dimension.

MovieLens 100K						
Model	dim=4	dim=8	dim=16	dim=32	dim=64	dim=128
FunkSVD	16	12	11	10	9	9
PMF	6	6	7	5	8	5
FM	7	7	5	4	3	3
AutoRec	1845	2375	1537	1558	1042	928
GMF	68	59	49	45	40	33
MLP	0	79	28	19	19	17
	80	61	51	44	37	36
NCF	32	0	0	14	16	1
	0	3766	5758	0	0	0
TransE	32	14	812	689	716	669
MovieLens 1M						
Model	dim=4	dim=8	dim=16	dim=32	dim=64	dim=128
FunkSVD	16	15	7	5	4	3
PMF	10	18	4	5	3	5
FM	26	33	12	6	4	3
AutoRec	3698	2161	3323	2951	1983	1451
GMF	507	442	251	213	175	150
MLP	167	176	141	139	232	-
	0	0	0	0	0	-
NCF	317	259	154	130	106	-
	17	131	122	128	106	-
TransE	32	0	1	569	1987	1345
Jester 617K						
Model	dim=4	dim=8	dim=16	dim=32	dim=64	dim=128
FunkSVD	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
PMF	1	1	2	2	2	0
FM	8	7	6	5	4	3
AutoRec	408	303	413	1124	81	117
GMF	40	36	11	9	8	7
MLP	25	30	24	24	0	3
	61	47	39	34	28	24
NCF	20	0	27	0	0	3
	0	0	0	0	0	0
TransE	160	175	291	195	223	270

Appendix C Indexing

Table A9: Indexing.

Recommendation (information or data)	Method	Model	Indexing	Typical works
Bipartite graph embedding-based recommendation (static user-item interactions)	Matrix factorization (MF)	FunkSVD and its variants	Sec. 3.1.1	[78], [78], [154]
		Combinations with KNN	Sec. 3.1.1	[156], [155], [157]
		Non-negative MF	Sec. 3.1.1	[158], [159], [160], [161], [162]
		Based on metric learning	Sec. 3.1.1	[170], [169], [174]
		Other factorization frameworks	Sec. 3.1.1	[175], [176], [177]
	Bayesian analysis	Automatic hyper-parameter adjustment	Sec. 3.1.2	[181], [183], [185], [188], [187], [163]
		Pair-wise ranking	Sec. 3.1.2	[190]
	Deep learning	For learning embeddings	Sec. 3.1.3	[193], [422], [196], [133]
		For proximity measurement	Sec. 3.1.3	[195], [199]
		Based on causal learning	Sec. 3.1.3	[191], [153]
	Others	Not missing at random assumption	Sec. 3.1.4	[205], [204]
		Positive-unlabeled problem	Sec. 3.1.4	[211], [212]
		Transfer learning	Sec. 3.1.4	[213], [221]
		Fast online learning	Sec. 3.1.4	[212], [225], [143], [226]
Bipartite graph embedding-based recommendation (temporal user-item interactions)	Matrix factorization	User's long-term preferences (online learning)	Sec. 3.2.1	[232], [233], [234], [235]
	Markov processes	User's short-term preferences (sequential learning)	Sec. 3.2.2	[238], [239], [238], [241], [242], [243]
		Automatic hyper-parameter adjustment	Sec. 3.2.2	[244]
General graph embedding-based recommendation (side information)	Translation	Techniques	Sec. 4.1.1	[251], [252], [67], [67], [253], [68], [255], [256]
		Recommendation models	Sec. 4.2	[293], [294]
	Meta path	Techniques	Sec. 4.1.2	[262], [264], [266], [69], [271], [265], [273], [274] [272], [270], [267]
		Recommendation models	Sec. 4.2	[295], [296], [297], [299], [423]
	Deep learning	Techniques of AutoEncoder	Sec. 4.1.3	[277], [278], [280], [281], [282], [284]
		Techniques of attention mechanism	Sec. 4.1.3	[287], [286], [290], [291], [292]
		Recommendation models	Sec. 4.2	[301], [302], [303], [304], [305], [307], [309]
Knowledge graph embedding-based recommendation (knowledge)	Large-scale learning	Techniques	Sec. 5.1	[336], [337], [338],[340], [265], [339], [286], [341], [342]
		Recommendation models	Sec. 5.2	[369], [370], [335], [371], [372], [373], [373], [375],[376]
		Recommendation models (meta path-based)	Sec. 5.2	[377], [299], [377], [72], [300]
	Multi-viewed graphs	Techniques	Sec. 5.1	[346], [347], [348], [345], [350], [351]
	Multi-layered graphs	Techniques	Sec. 5.1	[355], [356], [359], [272], [360], [361], [362], [363]
	Evolution	Techniques	Sec. 5.1	[364], [365], [366], [368]