Machine Learning for Refining Knowledge Graphs: A Survey

BUDHITAMA SUBAGDJA, D. SHANTHOSHIGAA, ZHAOXIA WANG, and
AH-HWEE TAN, School of Computing and Information Systems, Singapore Management University,
Singapore, Singapore

Knowledge graph (KG) refinement refers to the process of filling in missing information, removing redundancies, and resolving inconsistencies in KGs. With the growing popularity of KG in various domains, many techniques involving machine learning have been applied, but there is no survey dedicated to machine learning-based KG refinement yet. Based on a novel framework following the KG refinement process, this article presents a survey of machine learning approaches to KG refinement according to the kind of operations in KG refinement, the training datasets, mode of learning, and process multiplicity. Furthermore, the survey aims to provide broad practical insights into the development of fully automated KG refinement.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Computing methodologies** → **Machine learning**; **Knowledge representation and reasoning**; • **Information systems** → **Graph-based database models**; **Information integration;**

Additional Key Words and Phrases: Knowledge graphs, knowledge graph refinement

ACM Reference Format:
Budhitama Subagdja, D. Shanthoshigaa, Zhaoxia Wang, and Ah-Hwee Tan. 2024. Machine Learning for Refining Knowledge Graphs: A Survey. *ACM Comput. Surv.* 56, 6, Article 156 (February 2024), 38 pages. https://doi.org/10.1145/3640313

1 INTRODUCTION

Knowledge graphs (**KG**) have a rich expressive power to represent meanings and to enable complex reasoning over the represented knowledge. By representing entities in a domain as nodes and relations among the entities as directed edges, KGs have become the universal way to represent meanings in general. For any given domain, constructing a KG in terms of entities and relations commonly requires some ontologies, rules, or schemas to specify their semantics and meanings within the scope of the corresponding subject matter [1, 44]. The specification helps to ensure that the representation can be interpreted correctly, but still allows an automated reasoner to deduce additional information in the right context. In practical cases, especially in engineering domains, a KG may be used for multiple purposes, for example, supporting on-the-job training, augmenting information to relieve cognitive overload, and providing decision aids through question-answering

This research was supported by A*STAR under its Advanced Manufacturing and Engineering (AME) Programmatic Grant (Award No. A19E2b0098) and the Jubilee Technology Fellowship awarded to Ah-Hwee Tan by Singapore Management University.

Authors' address: B. Subagdja, D. Shanthoshigaa, Z. Wang, and A.-H. Tan, School of Computing and Information Systems, Singapore Management University, Singapore 178902,; e-mails: budhitamas@smu.edu.sg, shanthod@smu.edu.sg, zxwang@smu.edu.sg, ahtan@smu.edu.sg.

ACM 0360-0300/2024/02-ART156
https://doi.org/10.1145/3640313

interaction [142]. This may require semantics on concepts involving process, causality, laws of physics, analogy, and many others that are relevant to the domain and its strategy of presentation.

Issues and challenges in constructing knowledge graphs. Despite the ontologies and rules, KGs are always susceptible to errors and incompleteness during the initial stages of construction. The origin of the issues may be human mistakes when a graph is manually curated or some flaws in the algorithms or missing information in the data source for automated construction. After the construction, the KG still needs to be refined to fill in missing information and rectify inaccuracies. KG *refinement* [84] is known to be the process that adds missing information, corrects errors, and simplifies a graph by removing redundancies in an already existing pre-constructed KG. Some literatures consider KG refinement to be the same as KG *completion* [48] or KG *identification* [88]. However, Paulheim takes the KG refinement to cover all kinds of update to the KG, including completing missing information, detecting and correcting errors, and removing redundancies [84], whereas KG *completion* and *identification* cover only the addition of missing information and the identification of redundancies or errors, respectively. In this article, KG refinement refers to Paulheim's definition of the process to add missing information, correct errors, and simplify the KG [84] while KG completion is defined to be limited to the identification (or prediction) and addition of missing information (e.g., link prediction, relation prediction, and attribute completion) excluding the error correction or redundancy removal.

Traditional methods for refining knowledge graphs. Given the growing popularity of KGs in many domains, such as engineering [60, 129], biology [39], healthcare [67, 137], and social sciences [91, 120], the sheer size of information and knowledge to represent demands automation. This implies that the KG refinement requires the ability for a computer to complete missing information and correct mistakes on its own, with minimal or no human intervention. Traditional methods for automating the refinement process may involve the use of logical reasoning to deduce new information for completing or fixing the KGs [84]. In this case, powerful reasoning processes are required with very rich ontologies, rules, or logical axioms specifying the plausible patterns of contradiction, inconsistencies, and disjointing assertions among the entities in a KG [15, 72, 118]. However, as the graphs become very large with broad and deep material coverage, handcrafting reasoning rules, and heuristics to anticipate all possible erroneously conditions may also become enormously laborious and impractical [84]. To resolve this circular problem, one possible approach is to automatically acquire some models from data which can be used as the knowledge for guiding the process of refinement. The source of the data can be the KG itself, or can be from other sources, such as alignments with its domain ontologies or other KGs in related domains [96].

Machine learning for refining knowledge graphs. With the above motivation, various machine learning techniques have been applied to acquire various kinds of models from KGs [42]. The learned models enable various automated refinement tasks and operations, including completing, correcting, and simplifying the graphs. The techniques may range from traditional approaches in learning symbolic rules for detecting errors or anomalies [28, 80] to the popular deep learning methods that encode and process the KGs as neural networks [17, 42]. The learned models may also provide a more compact representation of the KGs, wherein the refinement process can be conducted more efficiently. For example, various approaches to representation learning have recently been applied to discover latent features in the graph. A low-dimensional vector representation can be formed so that it can be processed more effectively by most current methods of machine learning, especially those involving contemporary deep learning techniques (e.g., [12, 40, 65, 102, 116, 123]). This embedding step to map the graph into low-dimensional vectors representation becomes an important part of the process in most current KG refinement [133], especially those involving machine learning.

Machine learning for KG refinements can also be seen as composite, consisting of multiple approaches and learning techniques. Some refinement models have been regarded as autonomous agents acting to improve a KG. Under the reinforcement learning paradigm, the main actions of an agent are to explore, to complete, and to modify the graph such that the best way to refine the graph can be learned based on reward feedback from the environment (e.g., [37, 59, 114, 124, 126]). Here, the agent learns to improve a pre-constructed KG through exploration and trial-and-error, while the KG may undergo refinement along the way. In this context, it is interesting to study the mutual relationships between the process of learning how to refine the KG properly and the change made to the KG as the intended result of the refinement itself.

Objectives and the distinction of the survey. This article presents a survey of machine learning techniques and models that are used to support the KG refinement process. The techniques are categorized according to various factors, from the source of data for training, the learning paradigm (e.g., supervised, unsupervised, semi-supervised, reinforcement learning), and the multiplicity of the applied machine learning algorithms to the purposes of the outcome of the learning.

Although other surveys have reviewed different approaches to KG refinement and completion in general (e.g., [17, 84]), only a few have looked specifically at the involvement of machine learning in the refinement processes [4, 42, 62, 75, 122]. Those with emphasis on machine learning techniques in KG completion have mostly focused on representation learning and embedding techniques as the essential element that enables most modern machine learning techniques to process the graphs [4, 42, 62, 75]. On the other hand, some recent surveys with more comprehensive reviews provide a broader view on KGs, including their construction and refinement [122], quality management [127], and personalized interactions with humans [61]. However, they still consider individual machine learning techniques for particular tasks in processing the graphs, regardless of their roles in refining the pre-constructed KGs. In contrast, the survey in this article focuses on the use of machine learning for various types of KG refinement tasks, which may include completion, error correction, and graph simplification. In particular, the relationships between the knowledge learned by the machine learning models and the desired changes in the KGs as the main objective of the refinement will be investigated.

Based on our proposed KG refinement framework, a detailed account of how machine learning is applied to the particular type of KG refinement is provided. These applications may include combinations of different machine learning algorithms with different particular roles and interactions along the entire process. Notably, the detailed account of machine learning techniques used includes the emerging approaches of learning in this area, for example, reinforcement learning, which are intended to lead to fully automatic KG refinement.

The main contributions of this article can be summarized as follows:

(1) **A comprehensive overview of knowledge graph representation and refinement**. This survey introduces a broad technical overview related to the representation of KGs and the refinement process. The graph representations, ranging from the basic representation of nodes with edges to the latest emerging models of temporal knowledge, are introduced and defined. The process of the KG refinement is also defined as consisting of basic operations for analyzing and modifying an existing KG. These basic operations become the framework for characterizing different learning algorithms in this article.

(2) **A new taxonomy of machine learning models for the purpose of knowledge graph refinement**. This survey provides a new classification of machine learning models based on the classical learning paradigms (namely supervised learning, unsupervised learning, semi-supervised, and reinforcement learning) added together with some emerging types of learning techniques in the field of KGs (in particular representation learning). The taxonomy

includes practical characterizations of machine learning based on the training data availability (e.g., internal versus external to the initial KG, pre-trained models), particular roles in the KG refinement, learning process configuration (e.g., inputs, labels, observation, actions, reward feedbacks), and the possibility of continual refinement (e.g., offline learning, online learning, and incremental learning). Those characteristics may reveal the relationship between a policy, model, or knowledge acquired by a learning system and the update made to the KG as the result of the refinement process itself.

(3) **Practical insights and consideration for applying machine learning in knowledge graph refinement**. Different types of machine learning, as laid out in the taxonomy, are revisited with their particular roles in the basic operations of KG refinement. The following factors are considered: (i) Which kind of machine learning models can really be beneficial for a particular operation in the refinement? (ii) Are there still gaps in machine learning to support certain kinds of refinement operations? (iii) Which operations may be less affected by the lack of machine learning? These are discussed as practical considerations for applying machine learning in KG refinement.

(4) **Highlights and outlooks towards fully autonomous knowledge graph refinement**. A summary of the new categorization of machine learning and types of refinement operation is provided, with outlooks highlighting the future direction toward fully automated KG refinement.

The rest of the article is organized as follows: The overview of concepts related to KG refinement is provided in Section 2. Section 3 defines KG refinement as the term referring to the process of completing, correcting, or simplifying a KG. The section also presents a framework for characterizing the kind of operations involved in the refinement process. Section 4 provides an overview of machine learning models with their possible use in KGs. The section also provides detailed accounts of the survey on different machine learning models for KG refinement characterized based on the framework laid out in the previous section. Section 5 discusses the potential of realizing machine learning models in KG refinement and some gaps that still need to be filled by the current model of machine learning. Section 6 concludes the article. The organization of the article is also shown in Figure 1.

2 KNOWLEDGE GRAPHS

Graphs provide a way to assign meanings to a set of data. To express the meaning, entities of interest are represented as nodes in the graph, with relationships as directed edges between them. A KG is a graphic representation of knowledge that can express meaning or semantics. This section describes KG representation more formally from the basic vertices and edges, graph types and how they can be constructed. It also describes the main operations involved in KG refinement that may need machine learning to function effectively.

2.1 Knowledge Graph Representation

A KG is commonly represented as a directed graph

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \tag{1}$$

where $\mathcal{V}$ is the set of the graph vertex and $\mathcal{E}$ is the set of edge that links the vertices. A directed edge can be represented as a triple $(h, r, t) \in \mathcal{E}$ representing a relationship between two entities in $\mathcal{V}$, wherein $h \in \mathcal{V}$, $t \in \mathcal{V}$, and $r \in \mathcal{R}$ are the head vertex, tail vertex, and r as a distinct type of relation that connects the head with the tail, respectively, wherein $\mathcal{R}$ is the set of possible relation types of an edge in the graph. In a KG, a vertex $v \in \mathcal{V}$ represents an entity about a fact or a concept in the world associated with a particular meaning. In that case, the graph can be *heterogeneous* and

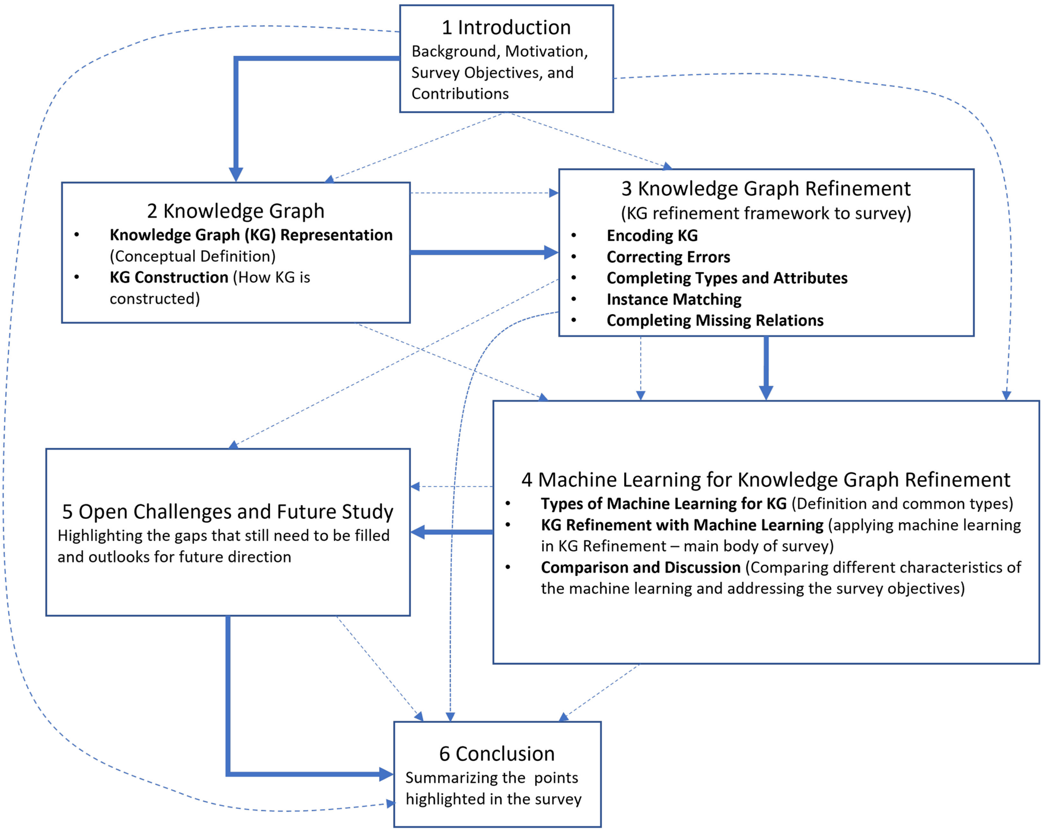
1 Introduction
Background, Motivation, Survey Objectives, and Contributions
2 Knowledge Graph
• Knowledge Graph (KG) Representation (Conceptual Definition)
• KG Construction (How KG is constructed)
3 Knowledge Graph Refinement
(KG refinement framework to survey)
• Encoding KG
• Correcting Errors
• Completing Types and Attributes
• Instance Matching
• Completing Missing Relations
4 Machine Learning for Knowledge Graph Refinement
• Types of Machine Learning for KG (Definition and common types)
• KG Refinement with Machine Learning (applying machine learning in KG Refinement – main body of survey)
• Comparison and Discussion (Comparing different characteristics of the machine learning and addressing the survey objectives)
5 Open Challenges and Future Study
Highlighting the gaps that still need to be filled and outlooks for future direction
6 Conclusion
Summarizing the points highlighted in the survey

Fig. 1. Organization of the survey paper in sequential order: Section 1 Introduction; Section 2 Knowledge Graph; Section 3 Knowledge Graph Refinement; Section 4 Machine Learning for Knowledge Graph Refinement; Section 5 Open Challenges and Future Study; and Section 6 Conclusion. Each section/topic may have subsections. The dashed arrows indicate content dependencies across sections.

the nodes and edges are enriched with additional information, such as labels, types, and attributes to imbue them with semantics and meanings. Every node may have a distinct *type*. For example, $\mathcal{V}_c \subset \mathcal{V}$ represent the set of vertex in the graph with type c.

To compute the graph representation, an adjacency tensor $\mathbf{M} \in \Re^{|\mathcal{V}|\times|\mathcal{R}|\times|\mathcal{V}|}$ can be used to represent the graph wherein a node corresponds to a row and to a column in a matrix in the tensor $\mathbf{M}$. Given that m_{irj} is a cell or an element at the ith row and jth column of an adjacency matrix of a sub-graph with all edges only of type r in tensor $\mathbf{M}$, $m_{irj} = 1$ if $\exists(i, r, j) \in \mathcal{E}$ or $m_{irj} = 0$ otherwise.

When the KG represents facts or states occurring in the real world, it may not be static over time. Edges and their types, labels, or attributes in the graph may be changing. There is an emerging field of temporal KGs attempting to extend the basic representation of KGs by imbuing additional information about the time when particular edges hold true (e.g., [56, 106]). Approaches to KG refinement that use the extended temporal graph representation will be discussed in more detail later.

Although the adjacency matrix can still be used to express basic graphs for computation, with possible extension to adjacency tensor to handle extra types, labels, or attributes as described above, there is still a scalability issue when the number of nodes and relations is quite large. The sparsity of the adjacency matrices and tensor demands high computation resources [17]. Another alternative representation is the *Laplacian* matrix, which summarizes important properties of the graph by representing the degree of connectivity of the corresponding vertex with its neighbors. In recent years, people have used KG embedding techniques to represent the graphs [115, 131]. Each entity and edge in the KG is translated into a vector of a low-dimension, producing a more efficient and dense representation that preserves the graph structure and semantics to be used for many different purposes. The survey in this article will discuss several types of embedding in more detail later, as parts of the new category of representation learning.

2.2 Knowledge Graph Construction

Constructing a KG involves the extraction of entities and relations from some external sources of data, such as natural language texts, web pages, and/or existing databases. Traditionally, the construction process can be done manually by domain experts, who handcraft every entity (node) and relation (edge) in the graph, together with their types, attributes, or labels [1]. Recent works on KG construction have been oriented toward automating the process [44].

Domain discovery, wherein data on the Web or in other repositories are mined to extract the entities and relations relevant to the domain or scope of the KG, can be regarded as the first stage in the automated construction process. Focus crawling can be employed to identify relevant sources of information (especially on the Web) to download for further analysis [44]. Based on the crawled and downloaded data (mostly in natural language texts), the next stage of the construction process is to extract the entity by identifying a token or a word referring to a name of a person, a location of an organization, an event, or another type of named entity through **named entity recognition** (**NER**) [2]. Further, the recognized entity goes through **named entity disambiguation** (**NED**) and **named entity linking** (**NEL**) to resolve ambiguities and to assign semantics with identifier, respectively [2]. Beside the entity extraction, relation extraction is intended to discover the semantic relationships between the entities extracted from the source text or information [1, 44]. KG construction is not in the scope of the review and will not be discussed further.

3 KNOWLEDGE GRAPH REFINEMENT

After a KG is constructed, it may contain incomplete, erroneous, and/or duplicated information. For example, some triples may be incorrect, redundant, or missing, so that the KG needs to be corrected, simplified, or completed, before they can be used in an application. The term *KG refinement* [84] thus refers to the process to achieve those tasks. Other literatures have also used terms, such as *KG completion* [48] or *KG identification* [88] to refer to the processes involved in KG refinement. However, they are mostly limited to the detection or identification of redundant entities or incorrect information, without specifying how a graph can be updated. In KG completion, the update of the graph may still be included but simply for addition of links or relations. This article makes use a broader scope of refinement that may include updates on completion and correction of the KG [84].

In contrast with KG construction, which may start from scratch and directly extract triples from texts, KG refinement starts with an already existing KG to fix or to expand. Based on various literature and related surveys [17, 42, 48, 84], several basic operations included in the refinement process can be identified according to the objective of the refinement tasks. Simplifying a KG requires a process called *instance matching* to find redundant information or duplicate entities in the graph [48]. The process involves updating the KG by removing the unwanted excessive information and merging redundant entities or linking them together to remark their similarity [48, 84]. A process for *correcting errors* needs to be conducted to identify and resolve possible incorrect or contradictory information in the graph [84]. In KG completion, missing information can also be identified and added through the process of *completing missing relations* to identify additional links establishing

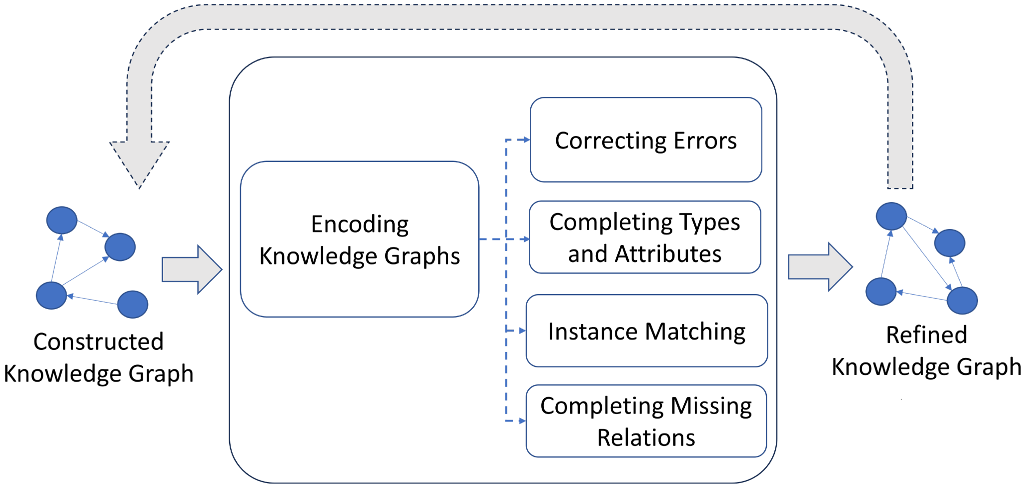
Constructed Knowledge Graph
Encoding Knowledge Graphs
Correcting Errors
Completing Types and Attributes
Instance Matching
Completing Missing Relations
Refined Knowledge Graph

Fig. 2. A framework for knowledge graph refinement. The dashed lines indicate the dependency between different operations.

new connections between nodes [17, 42, 84] and the process of *completing types or attributes* to fill up relevant types or attributes to individual nodes or relations [84]. To enable the refinement operations above, the KG needs to be represented in a form suitable for more efficient computing and analysis especially when the graph is large. *Encoding KGs* can be considered as an essential operation in the refinement process to encode the KG into a form suitable for more effective and efficient processing. Various encoding methods have been applied from the basic adjacency matrix, matrix Laplacian, to the recent graph embedding or latent space representation [17, 42, 48].

In this article, basic operations in KG refinement can be listed as follows: *encoding KGs*, *correcting errors*, *completing types and attributes*, *instance matching*, and *completing missing relations*. Each operation can be applied independently to address its own particular task objective. However, some or all of them may be combined to resolve the refinement task either sequentially or in parallel together. Multiple operations may also work together cooperatively to learn and refine the KG as described later in a section discussing reinforcement learning in KGs. Figure 2 shows the operations involved in KG refinement. Two or more operations may be used together to support each other in coming up with the overall tasks of refinement. For example, *correcting errors* can be made to run independently to clean the KG from errors, but it may also be applied together with *instance matching* to resolve redundancy and duplication from the KG. However, those downstream operations are mostly dependent on the result of the *encoding KGs* operation as it supports many inferences or predictions related to the representation of features in the KG.

In the following sub-sections, the operations are described in more detail.

3.1 Encoding Knowledge Graphs

Processes in KG refinement involve inferences and reasoning to extract features and properties of the graph representation relevant to identifying redundancies or missing information. *Encoding KGs* is an operation that aims to represent the knowledge graph in a form supporting such inferences to obtain the relevant properties of the graph. Traditional approaches typically encode the graphs into adjacency matrices (tensors) or spectral graph Laplacian representation suitable to compute the relevant node and graph properties, such as node *degree* (number of neighboring nodes), node *centrality* (structural importance of the node), *clustering coefficients* (clustered neighbors), the number of *graphlet*'s (sub-graph structure) occurrences, neighborhood overlap (node pairwise similarity), and optimal graph partitions [34]. Despite their potentials in providing useful feature information and powerful heuristics for KG refinement, traditional methods of encoding graphs with adjacency matrices and graph Laplacian are still inflexible. Extracting the relevant graph features and properties still requires handcrafted domain specific statistics and measures.

In recent years, as modern machine learning techniques such as deep learning have been gaining popularity, a lot of attention has been turned toward representation learning [6] including its application for encoding KGs. Representation learning in KGs can learn representations that encode the graphs' properties and structural information as neural networks. In particular, KG embeddings are types of encoding based on representational learning that express more meaningful information about entities and relations as low-dimensional, real-value vectors in continuous space. Representing them in a continuous vector space simplifies the operations involved to process the information while preserving the structural information about the KGs by translating elements of the triples into low-dimensional vectors. This can be useful for tasks, such as instance matching and relation prediction, as the embedding representation can be applied to obtain a score indicating the similarity between entities or the likelihood that a triple should exist in the graph.

More formally, this encoding operation can be defined as a pair of *encoder-decoder* function. The *encoder* maps each node in the graph into a low-dimensional vector representation, which can be defined as a function

$$F_{enc} : G \rightarrow \mathcal{M}^d, \tag{2}$$

where G and $\mathcal{M}^d$ are the set of graph representation and d-dimensional encoded graph representation, respectively. A basic encoding operation with F_{enc} may result in an adjacency matrix or tensor $\mathbf{M} \in \mathcal{M}^d$. A decoder function $F_{dec} : M \times \mathcal{M}^d \rightarrow \mathcal{F}$, on the other hand, must be configurable to extract some useful features from the encoded representation, where $M \subseteq \mathcal{M}^d$ be parts of the encoded graph representation. For example, given the entire adjacency matrix of a graph as a parameter, a decoder F_{dec} may be made to return all nodes' centrality of the graph as the returned features. In representation learning, the encoder and decoder functions are trainable to produce a representation and features suitable for the task at hand.

3.2 Correcting Errors

This operation identifies and fix errors in nodes or relations of the KG [84]. In this case, issues, such as errors, inconsistencies, and contradictions, are searched and corrected within the attributes of entities and relations. The main challenge in this error detection is to recognize incorrectness within the graph.

More formally, let $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ be an ideal KG that contains all the correct facts in the world without wrong or incorrect ones, where $\hat{\mathcal{V}}$ and $\hat{\mathcal{E}}$ are the set of node and edge (triple), respectively. Given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as the KG model to check. Let τ be a triple. The task of this operation is to identify whether $\exists \tau \in \mathcal{E}, \tau \in \hat{\mathcal{E}}$ or $\tau \notin \hat{\mathcal{E}}$. Accordingly, the head, relation, or tail element within τ can then be either kept or removed.

Besides factual truths or correctness, detecting errors, such as inconsistencies or contradictions, in the KG may require a powerful reasoning process with a rich ontology that defines what should and should not be true in the real world [84]. For example, PROSPERA (PRospering knOwledge with Scalability, PrEcision, and RecAll) has been made to cater to the ontological reasoning at that level by determining the plausibility of a new axiom based on the ontology [72]. This kind of reasoning needs disjoint axioms to state that a certain concept cannot be a member of two different classes.

Machine learning techniques have been applied to enrich ontology or background knowledge to support the reasoning about fallacies. A knowledge vault has been constructed using probabilistic methods to be used as a large-scale generic prior knowledge base supporting such reasoning [23]. Beside disjoint axioms, the prior knowledge can provide probabilistic information regarding the correctness of a relation. Another approach uses statistical distribution of triples directly on the graph to derive their correctness probability [86]. For a particular type of data, for instance, a numerical attribute, the error detection can also be done by statistically analyzing the graph itself for anomalies [119]. By looking at some deviations from common distributions, wrong or inconsistent assignments to numerical attributes can be detected. Similarly, error detection of numerical attributes can also be applied with probabilistic method, wherein a Bayesian learning is employed to learn the distribution [58]. Machine learning has also been applied to acquire rules to detect inconsistency in the entity attributes[66].

3.3 Completing Types and Attributes

This operation predicts and adds missing information as types or attributes of a node or a relation in the KG [84]. Predicting the types or attributes of an entity is commonly considered as a classification problem in which the existing information about a node or entity is mapped into a class or category indicating the type or group that share the same characteristics (to assign as new attributes). More formally, given an ideal KG $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ containing all correct facts including the types or attributes assigned to every vertex in it, let $v \in \mathcal{V}$ be a vertex in KG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are vertices and edges (triples) in $\mathcal{G}$, respectively. The task in this operation is to assign v to be in $\mathcal{V}_c \subset \mathcal{V}$ if $\exists\, \hat{v} \in \hat{\mathcal{V}}, \hat{v} = v$ and $\hat{v} \in \mathcal{V}_c$.

Traditionally, to infer the types of attributes that are unknown or missing, some entailment rules can be used as heuristics to map relation types or values as contexts to possible types of attributes [83]. This requires the rules to be complete and entirely correct. Although the heuristics can be improved by considering the statistical distribution of in-going and out-going relations [83], the approach is still domain-specific and may not be generalized to other domains. To tackle the limitations of handcrafting rules or heuristics, machine learning can be applied to automatically extract models and rules from data to fix the types and attributes in the KG.

To classify an entity (or relation), external sources of information, for example the Web, can be used to predict the type. The interlinks between Wikipedia pages can be learned using **k nearest neighbors** (**k-NN**) classifier to produce feature vectors to predict the type of an entity associated with a Wikipedia page in the KG [79]. Similarly, k-NN can be used to learn the features of different DBPedia language editions, so as to predict the missing types of the entity in the KG [82]. Others apply similar approaches with DBPedia abstract [29].

On the other hand, when entities do not have any association with an external source of information, supervised machine learning can be applied to the KG. Given the entities (with their representation of types and attributes) and the associated labels as types that come from the KG itself, a supervised machine learning algorithm can be employed to learn a classifier to predict the type given an input entity. A probabilistic method has been applied to acquire the conditional probability of a node to have a particular type given the relations that are connected to it [85]. **Support vector machines** (**SVM**) has also been used to assign entity types in DBPedia and Freebase based on the interlinks between KGs and the others' properties to predict the type [97]. Others use *matrix factorization* [77], association rule mining [83], and topic modeling [98] to predict entity types based on co-occurrences of other data items or related documents.

3.4 Instance Matching

Instance matching [48] is the process of finding and clustering instances or concepts in the KG to resolve them to be unique entities. This kind of operation has been known under many different names such as entity resolution, deduplication, data linking, or entity reconciliation [30]. The important part of this operation is to find that two or more entities are similar enough to be considered the same instances, so that they can be linked or merged together as a single entity. An entity can be duplicated by one or more other entities in the KG during construction, especially when the sources of the triples extracted are different, made at different times, or in different contexts. The duplicated entity may contain raw text attributes that may require some text processing to resolve, but it may also contain structured attributes that may or may not be aligned with the schema of the original one [71].

There are two main functions of instance matching: *match* function and *merge* function [8]. The match function is to indicate whether two entities or instances are the same. More formally, given node $v_1 \in \mathcal{V}$ and node $v_2 \in \mathcal{V}$, match function $\mathcal{M}(v_1, v_2)$ is true if v_1 and v_2 are the same instance (or false otherwise). $\mathcal{M}$ can be made to return a real value or made as fuzzy function returning a positive normalized value indicating probability or confidence that v_1 and v_2 are the same. On the other hand, the merge function may link, group, or merge the duplicated entities together. Once the nodes (v_1,v_2) are considered to be the same, a new entity that merges all attributes of v_1 and v_2 can be created and inserted into the KG. Depending on which node is dominant, the merge function may also be applied by removing either one of v_1 or v_2. In the linking data model of instance matching (e.g., [10, 27, 51, 74]), a new link between v_1 and v_2 is established to explicitly indicate that v_1 and v_2 are the same.

The basic model of instance matching traditionally uses the pairwise method to match the two entities by exhaustively obtaining every pair in the graph to match [3]. However, with a much larger KG, this pairwise method is considered inefficient. More recent models of instance matching have employed two stages of matching process. Firstly, *blocking* is a process that clusters similar entities using a lighter and less complex computation. Once blocking is done, a more rigorous pairwise *similarity* matching process is conducted within the group of matching entities [8].

Instance matching has been applied using handcrafted rules constructed based on some heuristics [3]. More recently, machine learning methods have been applied to realize the matching function [45–47, 78] either at the blocking stage, to rapidly group matching entities, or to more carefully merge duplicated nodes. Some models have also applied deep learning models [26, 53, 71] based on deep neural network architectures, which require the input data to be entirely vector representations. This implies that deep-learning-based instance matching has a dependency with the encoding KG operations to provide a low-dimensional vector representation for entities [26, 53, 71].

3.5 Completing Missing Relations

This operation aims to determine whether a relation exists between two entities of interest, given the types of the entities and the other relations [17, 48, 84]. If the relation should exist but still nonexistent in the KG, then the task includes filling up the missing one. The problem to solve in this case is to estimate the missing relations from triples. More formally, given an ideal KG $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ containing all correct facts without the incorrect ones. Let $(\hat{h}, \hat{r}, \hat{t}) \in \hat{\mathcal{E}}$ be a triple in $\hat{\mathcal{G}}$. The task of predicting relations is to determine whether $(\hat{h}, \hat{r}, \hat{t}) \in \hat{\mathcal{E}}$ or $(\hat{h}, \hat{r}, \hat{t}) \notin \hat{\mathcal{E}}$ given either element of $\hat{h}$, $\hat{r}$, or $\hat{t}$ is known. Let, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the target graph to refine. If $(\hat{h}, \hat{r}, \hat{t}) \in \hat{\mathcal{E}}$ and $\hat{h}, \hat{t} \in \mathcal{V}$ but $(\hat{h}, \hat{r}, \hat{t}) \notin \mathcal{E}$, then $(\hat{h}, \hat{r}, \hat{t})$ should be added to $\mathcal{G}$.

Traditional approaches to relation prediction uses pre-defined rules or statistical features to deduce the relation. For example, NELL is a rule-based system that operates based on the ontology of the KG to deduce new relations [15]. Similarly, KGRL uses inference rules to infer missing information in the KG [118]. However, the rule-based methods require a large number of rules and features to operate effectively.

Similar to completing types and attributes, relations in a KG can also be predicted using external sources of information such as the content on the Web. For example, patterns of textual keywords and phrases in Wikipedia abstracts can be learned using **conditional random fields** (**CRF**) to predict possible relations to different entities in a KG [55]. Using large text corpora from a particular source, such as Wikipedia, NER can be applied to link entities in the KG to the text corpus. Text patterns in the corpus corresponding to relations types can be acquired, and the patterns can be applied to predict additional relations from the text corpus [69]. When the KGs are interlinked with each other, the links can be used to identify and fill in the missing relations in one KG based on the information available in another one. This approach has been demonstrated using DBpedia, with multiple versions of different languages [14]. The idea of interlinked KGs has also been extended with probabilistic mapping between KGs that can be used to derive missing links in the KGs [25].

On the other hand, this operation can also be regarded as a classification task to predict the existence of relations. Some learning mechanisms can be applied to predict possible relations using the internal information of the KG itself. Meaningful chains of relations can be acquired for relation prediction through a process such as association rule mining [52, 83]. Prediction of the relation can also be improved by embedding the triples into vector representations to be processed further by a neural network architecture. Embedding of models, for example RESCAL[76], **Neural Tensor Network** (**NTN**) [99], and ComplEx [107] have been used for predicting relations based on the other existing relations in the KG. Another embedding technique maps pairwise entity-relations in Freebase into a lower dimensional space to predict the existence of relations [138].

Currently, various types of KG embeddings have been applied to predict relations. In recent years, much attention has been turned toward graph embeddings with representation learning especially with the advent of deep learning techniques in AI.

4 MACHINE LEARNING FOR KNOWLEDGE GRAPH REFINEMENT

4.1 Types of Machine Learning for Knowledge Graph

Machine learning is a process whereby a computer program constructs a model from data it has observed in order to improve its performance [70, 90]. Learning can be characterized according to the kind of improvements, the model it builds, and how the model contributes to the improvements. In the context of KG refinement, the input data to explore and discover the model may come from external sources, such as the Web and online databases [14, 23, 25, 29, 55, 69, 72, 79, 82]), but there are also learning algorithms that take the KG itself or part of it as the input data for learning. This also implies that the learning is conducted to explore the KG, which can be regarded as another kind of model that was previously constructed, in order to improve the representation of the KG.

This article adopts the common categorization of machine learning for KG refinement with the inclusion of a new type from the emerging topic of *representation learning*. The types of machine learning are as follows:

— *Supervised Learning*. Supervised learning is a kind of learning process that acquires the mapping from input to output based on samples of pairs of input and output data. Given training set $T = \{(i_j, o_j)\}_{j=1}^{N}$ wherein $o_j = f(i_j)$, a supervised learning process constructs a hypothesis function $h \in \mathcal{H}$ from T as an approximation or a model of the actual function f (ground truth). The result of supervised learning is usually applied for *classification* tasks wherein the domain of output sample o_j is a set of discrete classes or label C. Supervised learning may also be applied for *regression* tasks wherein the domain of the output sample is continuous or real $o_j \in \mathfrak{R}$.

— *Unsupervised Learning*. In contrast to supervised learning, which approximates the model based on list of pairs of input/output, unsupervised learning acquires the model based solely on the input data, without any output label. Instead of classifying the input based on predefined categories, the learning creates new categories or extracts new features for the given inputs without any label or output sample. Unsupervised learning is conducted based on training set $T^u = \{i_j\}_{j=1}^N$ consisting of only input data i_j without any label or output sample. The common purpose of unsupervised learning is *clustering*, wherein instances in the training set are separated into k clusters or groups.

— *Semi-supervised Learning*. This can be viewed as a type of learning between unsupervised and supervised learning. It is applied to learn the model when the labeled data are limited while the unlabeled ones may be abundant so that they are combined to come up with the best of both worlds. Supervised learning can be conducted firstly based on a few labeled samples followed by the use of the model to mine additional information from a larger set of unlabeled data [140]. This approach is related to the recent growing popularity of pre-trained models, especially for deep learning wherein a pre-trained model can be fine-tuned with unlabeled domain-related data instead of training it from scratch.

— *Reinforcement Learning*. In this type of learning, the instance of the machine learning program is considered as an agent residing in and interacting with an environment and receiving rewards at certain points to reflect how well it performs [90, 103]. The agent then learns to select the best action to take when it senses a certain environmental state, such that it will gain the maximum cumulative reward in the long run. An agent under **Markov Decision Process** (**MDP**) must take action $a \in \mathcal{A}$ when observing state $s \in \mathcal{S}$ according to policy π such that $a \leftarrow \pi(s)$, and it will receive reward $r \in \mathfrak{R}$ (can be positive or negative) while arriving at another state $s' \in \mathcal{S}$. In *policy-based* reinforcement learning, the policy can be parameterized with θ to return a conditional probability of action a, given s and θ as $\pi(a|s, \theta)$. The learning is to adjust θ to maximize its cumulative reward. In *value-based* reinforcement learning, such as Q-learning [117], the agent learns action-value function $Q(s, a)$ so that the policy becomes $\pi(s) = \arg\max_a Q(s, a)$. Both *policy-based* and *value-based* reinforcement learning can also be combined in an *actor-critic* model wherein one module learns in a *value-based manner* to guide (or criticize) the update of θ in the other module employing policy-based learning.

— *Representation Learning*. In this survey, *representation learning* is considered to be a distinct type of machine learning although it is similar to unsupervised learning wherein it learns from training data without labels. Representation learning encodes each instance data into a low-dimensional feature vector or matrix in order to obtain more useful information for further reasoning or learning tasks [6]. A neural network is enabled to learn the encoded vector representation so that similar input data get nearby points in the learned vector hyperspace [68]. The result of the learning (e.g., a language model in a neural network) can be used for prediction tasks.

4.2 Knowledge Graph Refinement with Machine Learning

Based on the framework of KG refinement as laid out in the previous section, different machine learning techniques can also be characterized according to the related operations for KG refinement. Figure 3 shows a taxonomy structure for characterizing machine learning to support KG refinement. The figure reflects the characteristics of different types of machine learning, according to the basic operations of refinement presented in this section.

4.2.1 *Supervised Learning in Knowledge Graph Refinement.*

In KG refinement, supervised learning is applicable to most types of operations. The training data can be parts or components of a KG, but the labels are usually provided from an external source or through manual input. Supervised learning may also be employed as the mode in representation learning for node classification and link prediction, with positive labels based on existing nodes or relations in the graph and some randomly generated contrastive data for negative samples.

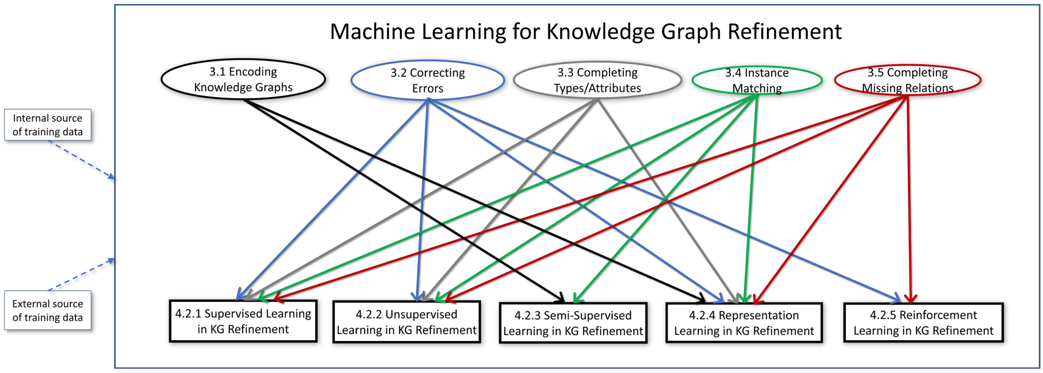
Machine Learning for Knowledge Graph Refinement
3.1 Encoding Knowledge Graphs
3.2 Correcting Errors
3.3 Completing Types/Attributes
3.4 Instance Matching
3.5 Completing Missing Relations
Internal source of training data
External source of training data
4.2.1 Supervised Learning in KG Refinement
4.2.2 Unsupervised Learning in KG Refinement
4.2.3 Semi-Supervised Learning in KG Refinement
4.2.4 Representation Learning in KG Refinement
4.2.5 Reinforcement Learning in KG Refinement

Fig. 3. A taxonomy of machine learning for knowledge graph refinement.

Supervised Learning in Correcting Errors. Very few KG error detection processes apply supervised learning, since the need for disjoint axioms may complicate the labeling and make it costly. Supervised learning has been used to train a probabilistic classifier in a knowledge vault model [23]. In this type of model, the input data for the learning is a triple from external sources, such as FreeBase. To automate the inclusion of disjoint axioms, the label is generated based on *locally closed-world assumption* heuristic to judge whether a triple is correct or wrong. The triple can also be assigned a probability value indicating to what degree it is right or wrong.

Supervised Learning in Completing Types and Attributes. Since predicting attributes and types of an entity can be considered as classification problem, it is straightforward to apply supervised learning for this kind of task. The k-NN classification learning has been applied to learn an entity in DBPedia interlinked with other Wikipedia pages to predict the type of the associated entity in a KG [79]. Typically, the k-NN algorithm is used to classify DBPedia entities based on the closest training examples in the labeled feature space of the training samples. Although k-NN was originally a clustering algorithm, the attached label in the input feature makes it a supervised learning one. The labels used in the identified case were derived from DBPedia ontology, such as place, person, organization, and activity. The labeled features can then be expressed as feature vectors wherein distances among different entities can be used to predict missing types in the graph. Similar approaches to using supervised k-NN model have also been adopted to learn entities from DBPedia language editions [82] and DBPedia abstract [29]. Another similar approach uses SVM to learn to predict missing entity types based on the links between different KGs when multiple KGs are involved [97].

Supervised Learning in Instance Matching. In instance matching, supervised learning has been applied to train classification systems to determine whether two different entities should belong together [9, 101] based on the similarity between the nodes' attributes and text features using learning algorithms, such as SVM, Bayesian learning, and Multi-layer Perceptron (Neural Network). The input data to the supervised learning model is the similarity representation of the pair of entities with a binary label indicating if the two entities in the pair should be considered as the same thing or as different entities. Recent approaches of instance matching involving deep learning models still also employ similar methods to train classifiers to predict if two entities are belong together in a supervised manner [26, 53, 71]. The difference is that the similarity measures can be directly obtained through the embedding vector of the entities. Supervised learning has also been applied for entity classification tasks, which can be considered as an important part of the instance-matching operation, by leveraging embedding representation such as GNN to include both content based semantic and structural similarity [21, 36, 50, 125]. In this case, multiple types of representation learning and supervised learning are combined.

Supervised Learning in Completing Missing Relations. Similar to the completion of types and attributes, completing missing relations can also be regarded as a classification task to predict the existence of relations. Some traditional methods use external sources of information or other KGs as training examples for relations or triples within the KG. For example, iPopulator completes the infobox display in Wikipedia with information linked to the textual contents of the corresponding Wikipedia page itself [55]. The training data are extracted from the texts of the first paragraph in the contents of the Wikipedia page as chains of features and labels. Each label is augmented with the position of labeled token in the article, representing the dependencies of the labels with other parts in the texts. Using supervised **conditional random fields** (**CRF**), the sequential dependencies among the tokens can be leveraged to predict the relations to populate the infobox. Another type of supervision in learning called *distant supervision* [69] has been used to train a relation predictor by taking the training data as entity pairs from the KG with the label extracted from other source (e.g., Freebase). By using the external generic purpose database, overfitting during the learning process can be avoided. When the KGs are interlinked with each other, the links can be used to identify and fill the missing relations in one KG based on the information available in another one. This approach has been demonstrated using DBpedia with multiple versions of different languages [14]. The idea of interlinked KGs has also been extended with probabilistic mapping between KGs that can be used to derive missing links in the KGs [25].

4.2.2 Unsupervised Learning in Knowledge Graph Refinement. Unsupervised learning has been applied in most types of the operation in KG refinement, especially when the training data required are large or sourced externally from the KG. It may also be more suitable when it is too expensive to manually label the training data for supervised learning.

Unsupervised Learning in Correcting Errors. Unsupervised learning has been applied for correcting errors in the KG itself. SDValidate has been used to provide a confidence score to each triple so that the triple can be assessed in terms of its correctness [86]. The correctness is derived from the **relative predicate frequency** (**RPF**) or the frequency of triple with the same predicate-object combination. KGist [5] is another unsupervised learning for detecting errors making use of KG inductive summarization. The learning summarizes the graph into inductive rules to indicate normal or abnormal conditions in the KG.

Unsupervised Learning in Completing Types and Attributes. Another example of unsupervised learning for completing types and attributes is SDType, which is based statistical distribution of types in the head and tail of a triple [85, 86]. Specifically, SDType measures the conditional probabilities of an entity to have a type T, given that the entity is in the head or tail position of the triple. Weights for every triple are applied to avoid the issues with skewed distributions when the population of certain types of entity is imbalance. The conditional probabilities can then be used to derive a confidence score for a type to be assigned to an entity.

Unsupervised Learning in Instance Matching. Unsupervised learning for instance matching is usually combined with supervised learning to make it semi-or weakly supervised learning (described in more detail in the next section). When it has to deal with large dataset containing many entities, the objective of unsupervised learning in instance matching is to acquire a model for grouping identical or similar entities in the quickest possible way before another round of rigorous matching process within each group. One interesting approach is the use of genetic algorithms to generate decision rules for matching entities when they should be grouped or linked together [78]. The decision rules may involve many aspects and parameters to set, such as similarity and comparable attributes. With a genetic algorithm, a population of the candidate "decision rules" is evolved with some selection and variation mechanisms towards the "fittest" solutions over many generations to converge the optimal ones. The "fitness criteria" is based on pseudo-precision and pseudo-recall when the candidate solutions are tested to obtain their fitness values.

Unsupervised Learning in Completing Missing Relations. The semantic relation composition approach has been used to extract first-order logic rules from the KGs for unsupervised learning for relation prediction [52]. The extracted rules are in the form of transitive relation among multi-hops of entities in the graph, so that a direct relation from the start to end of the chain of relation can be established. The rules are also quantified with support and confidence level. Another example of unsupervised learning in relation prediction is the probabilistic **case-based reasoning** (**CBR**) approach [19]. A model built on this approach learns the reasoning paths for predicting relations by gathering reasoning paths from similar entities in the KG. This probabilistic model estimates likelihood of effective path that answers query of a given entity. This CBR model uses a k-NN approach whereby it retrieves k similar entities from a query entity and detects multiple reasoning paths of each retrieved entity to a connected entity by query relation.

4.2.3 Semi-Supervised Learning in Knowledge Graph Refinement.

Semi-supervised learning in refining the KG is commonly used to manage a larger amount of training data whenever the labeling process is expensive. The learning may involve automatic generation of dataset labels, reducing the involved manual labors in training the system.

Semi-Supervised Learning in Encoding Knowledge Graphs. Semi-supervised learning can be applied in representation learning for the encoding KGs operation. Particularly, deep neural networks that are based on pre-trained language models, such as KG-BERT [132] and **Variational Graph Auto-Encoder** (**VGAE**) [36, 50, 125], as well as self-supervised learning may also be applied in encoder or decoder models in representation learning. In VGAE, the **graph neural network** (**GNN**)is trained to perform classification tasks. However, no external label can be provided to let the network learn directly in supervised manner. In this case, a semi-supervised learning approach can be leveraged in such a way that the learned labels can be propagated to other parts of training data that are still missing labels or just taken from the input as in the case of auto-encoder networks. Similarly, Transformer-based pre-trained model such as KG-BERT [132] can be used directly but often requires fine-tuning process. This fine-tuning usually involves self-supervised learning with a contrastive training regime wherein positive and negative training samples may be synthesized from the existing data set to enhance its discriminative capacity. A combination of contextualized embedding representation with structure-based graph embedding in relation prediction tasks can also be leveraged to handle actual problems in practice, such as to find candidate drugs to repurpose for COVID-19 [135]. A **literature-based discovery** (**LBD**) process is applied to seek and uncover valuable hidden connections between different research literatures regarding COVID-19 and to extract possible triples about the relevant concepts. The extracted triples are then constructed as a KG. Finding the candidate drugs can then be done by predicting links between entities in the KG through supervised representation learning to obtain translational, multi-relational, or contextualized (KG-BERT) embeddings.

Semi-Supervised Learning in Instance Matching. Semi-supervised learning has been applied for instance-matching operations, especially at the stage of blocking process that groups the same or similar nodes in what may be less accurate manner but is much quicker to process [45]. The learning algorithm runs in two separate phases. First, it generates a weakly labeled training set based on **term frequency and index-document frequency** (**TFIdF**) measures. The label can be negative (the nodes are dissimilar) or positive (the nodes are the same). Based on the KG as learned from the labeled dataset, the second phase of learning can start based on learned labeled data to refine the KG. Another approach makes use of *boosting* strategy, wherein a classifier is trained at first with some labels. In the second step, the learning continues with unlabeled training data [46, 47].

4.2.4 Reinforcement Learning in Knowledge Graph Refinement. In KG refinement, reinforcement learning is mostly used for reasoning about the path to traverse in the KG for question answering tasks or to predict missing relations between entities. The main difference from other types of learning is that the refinement program is considered to be an agent that interacts with the KG as its environment and learns to take some actions to update the KG. As a reinforcement learning agent, it receives feedback from the environment as reward signals, indicating its progress toward its task objectives. The state space observed by the agent commonly involves the current node position where the agent currently resides and may include the destination node as the target position in the graph. The action space wherein the agent can take to traverse the graph in every step is usually the set of possible relations that can be chosen to move in a single step from the current node position. The agent may receive the reward feedback when it reaches the target.

Reinforcement Learning in Correcting Errors. There are a few reinforcement learning models in KGs that are used for checking the validity of a triple. REL4KC [124] is a model of reinforcement learning for KG completion tasks. It is trained to learn paths in a KG from a structured database such as Wikidata to predict links or relations given an initial node and a target node to reach. With a policy-based learning approach and LSTM network to represent the history of states and actions taken, a policy is acquired as a probability distribution over possible actions that are sampled. Some extra negative samples from the external training data can be used to validate the triples in the KG and preventing false positive in prediction.

Another model called **generative reinforcement learning** (**GRL**) [114] uses an actor-critic model of learning together with LSTM to keep the historical information, but represents the graph in embedding space with a **graph convolutional network** (**GCN**) that aggregates many features and structural information of the graph in a multi-layer neural network. Besides visiting nodes and relations as the basic actions in the reinforcement learning, GRL may also generate new subgraphs to predict the paths. The generative process in GRL allows correct nodes to replace wrong or missing nodes from the original KG, and the GCN embedding makes it inferential, with more comprehensive awareness of the KG to judge the truth and falsity of the triples.

Reinforcement Learning in Completing Missing Relations. DeepPath is the first kind of reinforcement learning applied to the task of KG completion by predicting relations that can potentially answer particular questions [126]. An agent resides in the KG to traverse it to find the best path given a source and a target node. The agent's state space is the embedding vector representation, based on TransE [12] and TransH [116] embedding methods, of the node position where the agent currently resides and the distance to the target node position in the graph. The action space the agent can take to traverse the graph in every step is the relation that can be chosen in the embedding space from the current state. The agent will receive the reward feedback either when it fails to reach the target (as negative feedback) or a positive value reflecting path length and the diversity of choices. Since the number of possible actions to choose at one time can be very large, DeepPath applies policy-based learning to allow vector-based representation of action. This reinforcement learning model is also combined with supervised learning in which the agent learns some existing samples of successful paths during exploration to reduce the search space during learning. This also makes this reinforcement learning a case of a composite or multiple learning approach combining reinforcement learning, supervised learning, and representation learning (for the embedding representation).

MINERVA [18] is also a reinforcement learning architecture for KG completion with policy-based learning. Different from DeepPath [126], the learning in MINERVA is conducted from scratch without pre-training such as supervised learning to direct the exploration process. MINERVA also operates directly in the KG representation rather than the embedding space of the graph. Embedding representations may still be applied, but only for representing the observation vector of the agent. The observation space of the agent includes its current location in the graph, the relation connected to it, and questions to answer (for question-answering tasks). At every step, the agent chooses among outgoing edges connected to the current location. A positive reward is given only when agent reaches the target. To handle a large exploration space, MINERVA applies an LSTM model that encodes the history of state and the selected actions over time. Based on this encoded history, the policy chooses the action to take. Similarly, to handle sparse rewards in a large exploration space, M-Walk [94] uses a **Monte-Carlo tree search** (**MCTS**) with a customized **recurrent neural network** (**RNN**) to store the history of the search process. This M-Walk applies a value-based Q-learning model rather than policy-based reinforcement learning. **Rule-aware reinforcement learning** (**RARL**) [37] uses similar approaches with policy-based reinforcement learning and LSTM to deal with a complex exploration space, but applies a rule-based action selection strategy and beam search to direct the agent's exploration in the graph.

A more elaborated model employs a hierarchical reinforcement learning for KG completion [109]. Using a policy-based method, the so called **reasoning-like-human** (**RHL**) model conducts learning on a hierarchy of two-level reinforcement learning. The high-level policy in RHL is learned using policy-based learning that captures history vectors to keep historical information about the previous trajectory to better guide the agent. The state and action spaces at this level are similar to those in the other models of reinforcement learning, with positive rewards provided when the search ends up at the target. The low-level policy uses policy-based learning as well, but conducts it in the embedding space of the graph in different relation clusters that decompose a high level action into sub-actions. The state in the low-level policy consists of all valid sub-actions in the cluster, and a maximum positive reward is obtained when the agent reaches the next action in its high level policy. The embedding space is learned and represented with TransE [12]. In this way, the agent can learn to deal with multiple semantic meanings and contexts to reason and complete the KG.

Besides its application for validating facts in triples, GRL [114] can also be used for link prediction in KG. It uses an actor-critic model of learning together with LSTM to keep the historical information, but represents the graph in the embedding space with GCN, which aggregates many features and structural information about the graph in a multi-layer neural network. GRL employs hierarchical reinforcement learning as well, but unlike RHL [109], it generates a sub-graph using a **generative adversial network** (**GAN**). As parts of its action space, the agent can choose different actions such as continuing to walk in the KG or generating new sub-graph. The newly generated sub-graph is used to better predict the answer or the paths given the starting nodes. Similarly, REL4KC [124] can also be used for link prediction in addition to fact checking. In fact, the reinforcement learning agent is trained to perform link prediction using a publicly available external source of KGs, such as Wikidata. The rewards can be derived from embedding triple representation of the source KG, so that the agent will get additional reward value when taking the relation that leads to a true or correct fact compared to the invalid one.

MARLPaR is another model of reinforcement learning to resolve path reasoning problems such as query-answering tasks related to KG completion [59]. Two different reinforcement learning agents work together to explore the KG and accomplish the task. Each agent works similarly to the MINERVA agent [18]. However, one agent selects a relation to explore at every time step, while the other selects a node. A similar model of **multi-agent reinforcement learning** (**MARL**) has been applied [136]. One agent, called the KG reasoning agent, learns to answer a query in terms of a question triplet $(h, r, ?)$ in order to complete the missing element. Another reinforcement agent, called the information extraction agent, learns to rank useful knowledge from corpus in terms of ranked triples to help the KG reasoning agent to accomplish its task. Both agents receive a maximum positive reward when a correct answer is found. However, each agent has its own individual rewards, for example the minimum number of hops for the KG reasoning agent and the minimum number of unused triples.

4.2.5 Representation Learning in Knowledge Graph Refinement. In recent years, representation learning has gained considerable popularity as a method of encoding graph structures for KG refinement. It is known that tasks and operations, such as link prediction and instance matching, are directly supported by encoding the KGs in deep neural networks. Representation learning provides a representation that directly reveals distances between nodes, relations, and paths, or even the subgraph structure reflecting the semantics and structural relationships in the actual graph.

As mentioned previously, this type of learning involves an encoding phase wherein each data item to be represented, such as a node or a relation, is initially associated with a low-dimensional real vector. This vector can initially be assigned random values, or it can be based on some initial distributions. The learning process can then be conducted to train the decoder to adjust the values of the vector based on the conditions reflecting the distances between different data items in the embedding space.

These embedding models are commonly characterized based on an *encoder-decoder* architecture of a neural network for representing and processing the graph. The *encoder* maps each node into a low-dimensional vector representation, which can be defined as a function $f_{enc} : \mathcal{V} \rightarrow \Re^d$ where $\mathcal{V}$ and d are the set of vertices in the graph and the dimension of the vector, respectively. For example, $f_{enc}(v) = \mathbf{v}$ maps a node $v \in \mathcal{V}$ into its real vector embedding $\mathbf{v}$. A *decoder* function $f_{dec} : \Re^d \times \Re^d \rightarrow \Re^+$, on the other hand, predicts a value reflecting the relationship between two embedding vectors, such as the structural and similarity distance between the two.

There are various embedding models for KGs. *Individual entity* model is the simplest form of KG embedding wherein the embedding representation is applied to the content of each individual node or relation in the graph but not the structural information. In this model, the learning process is conducted similarly to the text-based or word embedding models. Since it does not include the structural information about the KG, the training data for the representation learning can be based on external sources in the form of textual documents or usage samples of particular words or phrases appearing in the KGs. These sample data may have also been formed as embedding representations of pre-trained neural network models wherein some major training phases may be skipped or simplified. This kind of embedding model often used in instance-matching operations [26, 53, 71].

Shallow embedding is another type of embedding model that represents individual nodes or relations of a graph in the embedding space as distributed vectors while considering other related neighboring entities in the graph to determine the nodes' (or relations') positions within the embedding space (e.g., [32, 87, 89, 104, 113, 134]). In this kind of embedding, the decoder function f_{dec} indicates the relative structural distance or *first-order similarity* between two entities in the graph reflecting the likelihood that one entity is connected or closely related (e.g., connected within a few number of hops or steps along multiple nodes) to another. This kind of shallow embeddings can be learned stochastically through random walks [16, 32, 87] and may represent the structural similarities between nodes (e.g., node2vec [32]), edges (e.g., edge2vec [112]), or subgraphs (e.g., Graph2vec [73]). *Second-order* similarity measures can also be obtained besides the *first-order* one. For example, in LINE [104] embedding model, the similarity of adjacency structure between two nodes can be obtained regardless the structural distance or position of them in the graph.

Translational embedding is another type of embedding representation wherein head or tail represents a point in the vast continuous vector space and the relation is a vector translation operation between them. The simplest form of this embedding, for instance TransE [12], follows a translation operation $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ for the encoder function, where $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t}$ are the vector representations of the head, relation, and tail, respectively. The decoder function $f_{dec}(\mathbf{h}, \mathbf{t}) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||_{L1/L2}$, provides the likelihood that the head and tail have a relation in the KG. Other variants of translational embedding, including TransH [116], TransR [63], TransD [40], and TranSparse [41], extend the embedding space with separate hyperplanes for nodes, relations, and among different types of relation, so that they can deal with more complex types of relations, namely reflexive, one-to-many, and many-to-one, that TransE alone cannot handle. To reduce the learning complexity due to the graph structural heterogeneity, TranSparse [41] has been used to handle heterogeneous and unbalanced KGs by leveraging sparse matrices that can optimize the computations required during learning. Learning the *translational* embedding representation require iterations of vector transformation and projection in different hyperplanes in the embedding space for every triple in the KG involving the embedding vector of a head node, tail node, and relation. Since the embedding involves the head, relation, and tail of the triple, it should reflect the actual structural or semantic distance as represented as the triple from its data source.

Beyond structural information, semantic correlation between nodes and relations can also be obtained with *multi-relational embedding* so that inferences, for example, analogical reasoning, can be applied to derive additional "unobserved" relations [7, 11, 64, 81, 100, 100, 128, 139, 139]. In this embedding model, each entity is associated with an embedding vector (e.g., based on word or phrase embeddings) with a matrix that represents the relationships to model the interactions between latent factors of the connected entities [11, 130, 139]. Extensions and simplifications have also been made in this type of embedding to reduce the space complexity of the network and improve the accuracy in handling both symmetric-asymmetric relations, bilinear tensor, and complex embeddings [99, 107, 130, 139].

A learning algorithm for multi-layer neural networks can then be applied, such as backpropagation or gradient-descent. The similarity between nodes can be based on binary value indicating the presence of the input relation (together with head and tail nodes) in the graph. It is also possible to provide a real or gradual value of similarity based on the similarity of the meaning between the head and the tail node when there is a way to externally obtain the semantic distance between them. The latter approach may also imply that the learning becomes supervised.

GNN is another class of deep neural networks that encode the KG based on a representation learning algorithm. In GNN, the neural networks represent nodes together with aggregated information from local neighbors. Based on the idea from **convolutional neural networks** (**CNN**), the node, with its aggregated local information, can be recursively extracted in a sequence of network layers. Deeper layers leverage the local features to construct highly expressive and more complex representations of the node both locally and holistically in the context of the entire graph structure. Variants of GNN [13, 22, 33, 35, 57, 92, 93, 105, 108] may use different ways on defining the neighbors, in obtaining the local aggregate information, and in processing the vector representation. In GNN, the representation learning process can be considered as a node receiving and aggregating messages continuously from its neighboring nodes from time to time in order to obtain the structural representation of the graph. More formally, let $\mathbf{h}_i^t$ be the embedding vector representation of node v_i at time t (it can be assumed $\mathbf{h}_i^t$ is a random vector at $t = 0$ or with the initial node features of v_i). For every iteration, $\mathbf{h}_i^t$ can be updated based on trainable weight parameters related to every connected neighboring nodes of v_i.

When the update applies, the node v_i may aggregate information not just about its own features but also about its neighboring nodes, which recursively include their neighbors, and so on. With a sufficient time, the aggregated information may cover the entire graph. For example, in GraphSAGE [35], as one of practical frameworks for GNN, each node is equipped with its own neural networks to process the aggregated features from its neighbors that can be obtained through averaging the vectors; filtering specific features; or even applying RNN (e.g., long-short-term memory or LSTM). Some techniques leverage more elaborate features and structure of the neural network architecture in which they are realized, and may employ different neural networks to realize the encoder and decoder functions. ConvE is a link prediction model based on a 2D convolutional network structure with multiple layers of non-linear features [22]. **Structure aware convolutional networks** (**SACN**) use **weighted graph convolutional networks** (**WGCN**) as encoder networks to combine translational embedding and link prediction as decoder networks [93]. Similar models with CNN structures combining different embedding layers, such as R-GCN [92] and RA-GCN [105], have also been proposed. Sequential learning with RNN has also been used to learn the local embedding features as if they are coming in a sequential order. The deep sequential model for KGs (DSKG) aggregates information about the node features as a sequence learning in RNN wherein entities and relations are presented in sequence [33]. LSTM has also been applied to obtain sequence embeddings combined with attention mechanism [57]. Graph attention network or GAT applies attention mechanism for processing local information from the neighbors [108].

Most operations in KG refinement can be simplified by utilizing the encoded KGs from the representation learning. Inferences, such as validating triples and predicting missing nodes/relations, can be conducted by the decoder function in the neural network, wherein the graph is represented as the neural network itself to obtain the results. However, as machine learning models, these KG encoding must be trained to find the right configuration of distributed vector representation in the multi-dimensional hyperspace before the proper refinement process can be conducted. The KG representation may also impose some challenges to representation learning including training complexity, graph heterogeneity, and dynamic temporal representation. The requirements to visit every node, relations, or combination of triples during training may make the process in representation learning quite complex. Many representation learning models employ stochastic methods to visit different nodes in the graph (e.g., random walks) to learn about their properties and relationships, but mostly assume homogeneous structure of the graph wherein types of relation in the graph are indistinguishable. Graph representation learning with this homogeneous assumption may not be effective to apply to heterogeneous graphs as significant information from different types of nodes or relations may be ignored or missing. On the other hand, special treatment may be required to handle KG with changing features or properties over time or require time information to be explicit. More details on how existing approaches address the challenges will be given in the following parts of this section.

Representation Learning in Correcting Errors. In KG-BERT [132], self-supervised learning is commonly applied for the triples with contrastive learning. The pre-trained network is fine-tuned with positive and negative examples synthesized from the training data for encoding the KG. This allows the representation learning in KG-BERT to learn to validate whether a triple should or should not exist in the KG. This fine-tuning of a pre-trained language model involves different sources of training data. Firstly, the pre-trained language model is considered to be initially trained on an external source of free text data. Secondly, the triples to be represented as token sequences as the input data are obtained from the KG itself. Lastly, the positive and negative labels for the input triples are derived from the triples themselves. A positive label indicates that the triple really exists in the KG, while a negative one states that it should not be existed. The negative triple data can be generated by randomly replacing the head or tail entity in one of the positively labeled triples.

In GRL [114], the GCN aggregates many features and structural information of the graph in a multi-layer neural network. The GCN makes the agent inferential, with a more comprehensive awareness of the KG to judge the truth and falsity of triples. This allows a GRL agent to generate sub-graphs and to replace wrong or fill up missing nodes in the original KG. In this case the encoded graph in the form of GCN must be leveraged by a GRL agent to enable the complete operation of *correcting errors*.

Translational embeddings (e.g., [12, 40, 41, 63, 116]) can also be leveraged as simpler ways for *correcting errors* in KG through a *triplet classification* task. The task is a binary classification to determine if a triple (head, relation, and tail) is correct or not, based on the score obtained directly from the embedding representation. This approach, however, still requires the process of checking every possible triple in the graph to detect the incorrect ones and limited to errors in relations only.

Representation Learning in Completing Types and Attributes. Multi-relational embedding models allow nodes' and relations' properties to be semantically correlated. In this way, it is possible to use the learned model to complete the KG at the node attributes and properties levels. RESCAL [76], as a multi-relational embedding model that learns correlations among the elements in triples using a three-way tensor factorization, has been extended to include node attributes information [77]. The information about attributes is added as an entity-attribute matrix factorization during the embedding representation of the attributes. This matrix is included in updating tensor $\mathcal{X}$ during learning so that the correlation among nodes, relations, and attributes can be obtained. Predicting attributes is possible, in this case, to approximately determine if a particular triple with given attributes should exist. This also implies that the multi-relational embeddings can be trained to conduct regression tasks that predict or generalize new information beyond the training data.

Representation Learning in Instance Matching. DeepER uses word embeddings to learn the semantic relationships among nodes in the graph [71]. The system predicts if each pair of entities is matched or mismatched. Similarly, Deep Matcher is also based on a pairwise entity classification task of node properties with logistic loss [26]. DeepNN-ER also uses word embeddings as the pre-processing [53]. These pairwise classification tasks require supervised learning after the embedding vectors for every node are produced. However, these semantic embeddings [26, 53, 71] are still considered *individual entity* embeddings, as they do not include structural information nor relations among the entities in the context of the graph. This also implies that the learning processes involved are limited to producing text-or word-level embedding representation with training data obtained from some external sources (e.g., publicly available corpus, source text documents) rather than the KG itself.

The task of instance matching can also be seen as a node classification problem in that two nodes can be considered to be the same if they belong to the same class or group. VGAE is a kind of GNN embedding model that can be trained for node classification tasks [36, 50, 125]. A self-supervised learning is applied to enable the network to classify the node. In this case, the classification takes the structural aspect of the nodes into account as well. The task of node classification based on the structural aspect of the graph can also be achieved by a *shallow embedding* or *skip-gram* method of encoding [32, 87, 89, 104, 113, 134]. However, when the KG is heterogeneous, direct stochastic methods for visiting the nodes may not be effective and the results may potentially be biased towards those with a dominant number of paths. Metapath2vec [24] can applied for node classification tasks for heterogeneous KGs by generating meta-path as guidance for random walks during representation learning such that semantic relationships between different types of nodes can be properly incorporated into the embedding representation. Although, both GNN and translational embeddings do not learn to perform the overall instance matching processes, it is straightforward to obtain the score indicating the similarity between two different nodes in the graph. However, some exhaustive processes may still be needed to search all possible pairs of nodes to find the best match and to determine if they are belong to the same instance.

Representation Learning in Completing Missing Relations. Most, if not all, encoding models with representation learning mentioned in this article represent nodes as vectors in the low-dimensional embedding space. It is straightforward to make use of these embedding representations to predict either existing or missing relations. Given two vectors of different nodes, distances between them can be obtained, which predict how closely they are related. All the *shallow* [32, 87, 89, 104, 113, 134]), *translational* [12, 40, 63, 116], and *multi-relational* [7, 13, 32, 64, 81, 100, 128, 139] embeddings mentioned previously can be used to obtain the likelihood that two different nodes in the graph should be connected either structurally or semantically. Some multi-relational and GNN [13, 22, 33, 35, 57, 92, 93, 105, 108] models can also be used to predict new possible relations even though they do not exist in the original graph as the internal source of training data set for the embeddings. These embedding techniques can also be used for link prediction, together with another learning paradigm, for example reinforcement learning [109, 114, 126]. Some models of representation learning for link prediction are also made to deal with heterogeneous KGs. For example, ComplEx [107] is a multi-relational embedding model that leverages *complex* vectors to deal with the graph heterogeneity issues. **Heterogeneous Graph Transformer** (**HGT**) [38] is a GNN leveraging meta relations that parameterize weight matrices for calculating attention for every relation to represent heterogeneous graphs. KGTtm [43] leverages crisscrossed neural network to measure trustworthiness of triples which can be applied for predicting relations.

Besides the embedding models, there are more examples of combinations of representation learning methods and applications for predicting relations. **Structured-augmented text representation** (**StAR**) learning [110] applies a hybrid model of textual encoding of triples with KG-BERT [132] and transitional graph embedding, such as TransE [12] and RotatE [102], to combine the capabilities of the contextualized sequential representation with structural features in the embedding model. A Siamese-style textual encoder is applied to the triples of the contextualized representation while applying a different scoring module to the two in order to get the best of both contextualized and structural representation. Self-supervised learning is applied in this case, with labels indicating if a relation exists between two nodes in the graph. If so, positive examples are generated. Contrastive learning is also used by randomly changing entities in the triples to generate negative examples.

Another method with KG-BERT is to use multi-task learning to identify more relational properties in the KG [49]. This is achieved by combining the objectives of relation prediction and relevance ranking in the learning for link prediction. A multi-task learning method was created to learn more relational properties in the KG. A combination of contextualized embedding representation with structure-based graph embedding in relation prediction tasks can also be leveraged to handle actual problems in practice, such as to find candidate drugs to repurpose for COVID-19 [135]. A **literature-based discovery** (**LBD**) process is applied to seek and uncover valuable hidden connections between different research literatures regarding COVID-19 and to extract possible triples about the relevant concepts. The extracted triples are then constructed as a KG. Finding the candidate drugs can then be done by predicting links between entities in the KG through supervised representation learning to obtain translational, multi-relational, or contextualized (KG-BERT) embeddings. As learning models for predicting relations, supervised learning approaches with contrastive labeling are also applied in representation learning to create combinations of contextualized and structural embeddings.

Predicting relations can also be made inductive such that the model can infer a new triple with head and tail entities that are hidden or not present in the original KG. In a **commonsense knowledge graph** (**CKG**), wherein entities are composed of free-form text, InductivE [111] has been used as a framework for conducting inductive KG completion. Given the initial form of CKG, a graph densification process is applied to generate high-quality links between semantically similar entities during the graph encoding process. Using a graph convolution network as the encoder and ConvE model as the decoder, InductivE can predict unseen or hidden triples in the KG. In this case, contrastive labeling in supervised learning is also used to enable the prediction.

Beyond simple geometrical structures to explore in translational embeddings, a few variants have been made to consider more complex relationship between entities. OPTransE [141] extends TransE's translational KG completion to take relational paths into consideration. Unlike the other translational models, OPTransE projects the head and tail of a triple into different embedding spaces. In this way, it can keep the relational order information beyond just a collection of relations along the path.

Hierarchy-aware knowledge graph embedding (**HAKE**) [137] is another variant of translational graph completion that encodes semantic hierarchical structure among the entities. The semantic relationships, as described by relations in the triples, are encoded in a polar coordinate system, such that the radial coordinate indicates the entity level in the hierarchy while the angular coordinate represents the semantic distance within the same level.

One of the issues in representation learning for KG completion in general and link prediction in particular is the complexity of the training process, which requires possible combinations of relation triples in the graph to be presented so that missing links can be predicted. Together with contrastive supervised learning, few shot learning techniques have been proposed to deal with the issue. These techniques allow the learning model to take only few reference triples in order to come with more complete predictions. Adaptive attentional network for few-shot KG completion (FAAN)[95] has been used to learn to predict links and relations based on a few number of reference samples to learn. The reference triples are encoded adaptively with translational embedding and attention mechanisms in transformer networks to match the references with queries. **Few-shot relation learning** (**FSRL**) [137] is another model of few-shot learning for KG completion using attention mechanisms as well, but applies GNN to aggregate relations of the reference triples with recurrent autoencoder.

Besides the complexity of training a link prediction system, another issue with a temporal KG is that every relation is associated with temporal information, so that a triple may be true for a certain time, but does not hold in another. Moreover, the KG may also change over time, so that the representation learning must also be incremental. A straightforward approach to tackling this is to apply *diachronic embedding*, wherein the entity and timestamp are taken for the embedding [31], so that contrastive supervised learning can be applied to learn for link prediction. Using SimplE as translational embedding with diachronic model, DE-SimplE [31] is a model for temporal KG completion based on diachronic embedding. TComplEx and TNTComplEx [54] are extensions to ComplEx [107] multi-relational embedding to include timestamp attributes in the embedding space. HyTE [20] is another temporal extension of translational embedding that incorporates temporal information directly to hyperplanes in the embedding space. In this case, the task becomes temporal link prediction as in TComplEx and TNTComplEx [54] that amounts to predict the head, tail, or relation held at certain time as specified in the timestamp [54]. However, incremental learning remains an issue in this kind of diachronic embedding. **Time-aware incremental embedding** (**TIE**) [121] has been proposed to deal with the incremental learning issues and the dynamic of the KG. Instead of just simply applying diachronic embedding, TIE makes use of experience replay techniques to maintain updated relation in KG and to avoid catastrophic forgetting. In that case, facts from new time steps can be incorporated while preserving knowledge derived from the previous ones. TIE selects pattern frequencies among samples and only uses deleted and added facts at the current time step for training to improve performance.

4.3 Comparison and Discussion

From the different types of machine learning for KG refinement described above, it is clear that there are classical models of machine learning directly applied and operated on the KG itself and there are more modern approaches that use embedding representation of the KG to perform the operations in KG refinement. Table 1 summarizes different machine learning models applied to the encoding KGs operation. Table 2 shows the other machine learning applied to the other four operations. Each column in Tables 1 and 2 can be described as follows: *Refinement operations* (*Refine op.*) indicates the KG refinement operation; *Learning type* indicates the type of machine learning (supervised, unsupervised, semi-supervised, reinforcement learning, or representation learning); *Training data* describes the source of the training set used by the machine learning (can be *internal* from the KG itself or externally from other sources); *Multiplicity* indicates whether more than one (single) learning algorithms are applied to resolve the same problem; *Supported operations* indicates what KG refinement operations are supported by this machine learning approach besides the one indicated in *Refine op.* column; and *Description* provides some detail about the learning approach. Based on the kind of refinement operation, type of learning, source of input, other operations supported in the refinement process, and the multiplicity of the learning algorithm, it is also indicated that several machine learning approaches for different KG refinement operations involve different machine learning algorithms. One learning approach may also support multiple KG refinement operations.

4.3.1 Classical (Non-embedding) Machine Learning for Knowledge Graph Refinement.

Most classical approaches to machine learning operate directly in the KG representation to improve the entire refinement operation. They cover most of the operations which includes updating the KG as the target of the refinement. For example, classical approaches to learning for instance matching [9, 101] involve merging or link establishment to the KG after the nodes that should be under the same category or should be together are identified. Similarly, unsupervised or semi-supervised learning for instance matching [45–47, 78], error detection [23, 85], type completion [29, 79, 82, 86, 97], and relation prediction [14, 19, 25, 52, 55, 69] involve the required changes to the KG as parts of the refinement objective. Depending on the algorithms, some of them learn internally or directly from the KG itself as the target of the KG refinement [9, 85, 86, 101], but most other may take some external sources of data to be used as the training set [14, 19, 25, 29, 45–47, 52, 55, 69, 78, 79, 82, 97]. These external data may come from free text documents [14, 25, 55, 69], publicly available databases [29, 79, 82], or as populations of rules [97]. In some operations such as correcting errors and completing missing relations, the required updates in the KG are directly supported by reinforcement learning [109, 114, 124] wherein the actions to change the graph are inherent in the methods. However, the design of the learning refinement system must incorporate reward mechanisms to direct the agent behavior towards the task objectives.

4.3.2 Modern (Representation) Machine Learning for Knowledge Graph Refinement.

Most modern approaches of machine learning for KG refinement that leverage representation learning for encoding the graphs are commonly considered only for detecting or identifying the duplication, errors, or gaps to fulfill. No further updates on KG are commonly applied, except for a direct

Table 1. Characteristics of Machine Learning in *Encoding Knowledge Graphs* Operation for Knowledge Graph Refinement

Refine op.	Learning type	Algorithm/Model	Training data	Multiplicity	Supported operations	Description
Encoding KG	Rep. Learning	Shallow-Embedding: Node2Vec, DeepWalk, NetMF, LINE, SDNE, LBD, MetaPath2Vec [24, 32, 87, 89, 104, 113, 134]	internal	single	Encoding KG, Instance Matching, Completing Missing Relations	Random walk and Matrix Factorization to learn the embedding vectors
		Translational-Embedding: TransE, TransD, TransR, TransH, TranSparse [12, 40, 41, 63, 116]	internal	single	Encoding KG, Instance Matching, Completing Missing Relations, Correcting Errors	Vector/matrix transformation in embedding space
		Multi-Relational-Embedding: RESCAL, NPL, Spectral-Network, ANALOGY, LRE, NTN, HolE, ComplEx, KGTtm [5, 7, 21, 43, 64, 76, 81, 100, 128, 139]	internal	single	Encoding KG, Instance Matching, Completing Missing Relations, Completing Types & Attributes	Matrix Factorization to handle structural and semantic information
		GNN: Spectral, ConvE, DSKG, GraphSAGE, PathBased, SACN, RA-GCN, GAT, HGT [13, 22, 33, 35, 38, 57, 92, 93, 105, 108]	internal	single	Encoding KG, Instance Matching, Completing Missing Relations, Correcting Errors	Learning and embedding in terms of Graph Neural Networks
		Individual-Entity-Embedding: DeepER,DeepMatcher, DeepNN-ER [26, 53, 71]	internal	multiple (supervised learning, Rep. Learning)	Encoding KG, Instance Matching	Word embedding to learn semantic among nodes in the graph
		Translational-Embedding-and-GNN: InductivE, OPTransE, HAKE, FAAN, FSRL [95, 110, 111, 137, 141]	external (common-sense KG, freetext)	single	Encoding KG, Completing Missing Relations	Inductive learning in GNN (InductivE) and translational embedding
		Temporal-Embedding/Diachronic: DE-SimplE, TComplEx/TNTComplEx, HyTE, TIE [20, 31, 54, 121]	internal	multiple(Rep. Learning, memory-based/ experience-replay(TIE))	Encoding KG, Completing Missing Relations	Incorporating temporal information in the embedding space, leveraging experience replay to preserve time information (TIE)
	Semi-Supervised	GNN: KG-BERT [132]	external (pre-trained language model)	multiple (Rep. Learning, semi-supervised learning)	Encoding KG, Completing Missing Relations, Correcting Errors	Fine-tuning pre-trained language model with self-supervised learning and contrastive labeling
		GNN: VGAE [36, 50, 125]	internal	multiple (Rep. Learning, self-supervised learning (contrastive))	Encoding KG, Instance Matching	Node classification to check node similarity
		GNN: StAR, Context-Struct relational, LBD (Covid-19) [49, 110, 135]	external (pre-trained, freetext)	multiple (Rep. Learning, self-supervised learning (contrastive))	Encoding KG, Completing Missing Relations	Fine-tuning pre-trained language model with self-supervised learning and contrastive labeling

update, such as link prediction that only requires additional links or relations to any pair of nodes in the initial KG [31, 49, 95, 110, 110, 111, 121, 135, 137, 141] or those with *individual-entity* embeddings [26, 53, 71]. For *individual-entity* embeddings, the representation learning is only used to obtain the embedded vector space of individual nodes to identify whether some of them can be

Table 2. Characteristics of Machine Learning in *Correcting Errors*, *Completing Types and Attributes*, *Instance Matching*, and *Completing Missing Relations* Operations for Knowledge Graph Refinement

Refine op.	Learning type	Algorithm/Model	Training data	Multiplicity	Supported operations	Description
Correcting Errors	Supervised Learning	Knowledge Vault [23]	external (Freebase)	single	Correcting Errors	Learning a probabilistic classifier from triples
	Unsupervised Learning	SDValidate [85]	internal	single	Correcting Errors	Statistical learning to provide confidence score to each triple
		KGist [5]	internal	single	Correcting Errors	Knowledge graph Inductive Summarization to extract soft rules
	Reinforcement Learning	Rel4KC [124]	external (wikidata)	multiple (RL, Supervised Learning, Rep. Learning)	Correcting Errors, Completing Missing Relations	RL (policy based with LSTM).
		RHL [109]	internal	multiple (two RL instances)	Correcting Errors, Completing Missing Relations	Hierarchical RL, policy-based (high-level), policy-based on embedding (low-level).
		GRL [114]	internal	multiple (RL, Rep. Learning, generative (GAN))	Correcting Errors, Completing Missing Relations	RL (actor-critic with LSTM), RpL (GCN), GAN.
Completing Types_& Attributes	Supervised	*k*-NN [29, 79, 82]	external (DBPedia, Wikipedia pages)	single	completing types_& attributes	Learn to classify the types of entities.
		SVM [97]	external (multiple KGs)	single	completing types_& attributes	Learn to predict missing entity types based on links between different KGs
	Unsupervised	SDType [86]	internal	single	completing types_& attributes	Learn conditional probabilities of an entity to have a certain type
Instance Matching	Supervised	SVM, Bayesian, MLP(NN) [9, 101]	internal	single	Instance Matching	Matching based on the similarity between nodes' attributes and text features
	Unsupervised	Genetic Algorithm [78]	internal	single	Instance Matching	Genetic algorithm to generate decision rules for matching entities
	Semi-supervised	TFIdF [45], boosting [46, 47]	external (initial labeling)	multiple (supervised followed by semi-supervised learning)	Instance Matching	Two separate phases: generates labels with TFIdF and learn based on labeled data
Completing Missing Relations	Supervised	iPopulator, CRF-dist.supervision, interlinked KGs, prob. interlinked KGs [14, 25, 55, 69]	external (Wikipedia, Freebase, DBPedia, multiple KGs)	single	Completing Missing Relations	Supervised learning with distant supervision: training data and labels taken from different KGs
	Unsupervised	Semantic relation composition [52], Probabilistic CBR [19]	internal	multiple (CBR and *k*-NN in Probablistic CBR)	Completing Missing Relations	Learn rules and reasoning paths as hops in KG
	Reinforcement Learning	DeepPath [126]	external (samples of successful paths for supervised learning)	multiple (Supervised, Rep. Learning, RL)	Completing Missing Relations	Learning from reward and from successful path samples
		MINERVA, M-Walk, RARL [18, 37, 94]	internal	single	Completing Missing Relations	Learning to predict relation and path reasoning
		MARLPaR [59], MARL [136]	external (text corpus for agent 2)	multiple (multi-agent RL)	Completing Missing Relations	Two agents work together to resolve path reasoning task

grouped together, independent from the original KG. In that case, the change to the graph can be directly applied.

Other embedding-based methods (e.g., [21, 36, 50, 125, 132, 132]) that require updates to the KG may also constitute costly retraining of the embedding representation. Changes in node or relation properties, removal of entities or relations, or triples may require changes in the embedding representation, as well as relearning the representations for all elements in the graph, with the exception of GNN embedding models (e.g., [21, 36, 50, 121, 125]). In GNN embeddings, changes in neighboring nodes' properties can still be learned incrementally with the aggregating process of neighboring features. Although the process may still be expensive [35], updating the embedding representation of the KG may not be needed, as the graph change can still be considered as part of the input data to the neural network.

Table 1 shows that the application of representation learning for KG refinement is dominated by operations, such as relation prediction and node classification, which are important for instance matching or entity resolution. Few approaches to embedding representation are also applied for detecting errors [132]. However, they still need to be combined with other kinds of machine learning, pre-trained models, or contrastive learning from other validated sources. The embedding of representation as the results of representation learning in the modern approaches can also be multipurpose, so that besides predicting relations, it can also be applied for instance matching, detecting error with classification, and completing types or attributes.

4.3.3 Integrated Processes in Knowledge Graph Refinement: Reinforcement Learning. Some relatively recent machine learning models such as reinforcement learning have been applied to KG refinement with some independence from embedding representation learning. Tables 1 and 2 summarize different machine learning models applying reinforcement learning and representation learning, respectively. Reinforcement learning demands incremental changes to the KG as the agents explore and perform some actions in the graph as their environment. Some reinforcement learning approaches operate directly on the KG representation itself, without the need for embedding representation [18, 37, 94]. The approaches are made with the improvements of DeepPath [126] as their predecessor. DeepPath makes use of embedding representations of the KG instead of the KG itself, and requires some combination with supervised learning to direct the exploration toward favorable outcomes for every action step. Those that operate directly on the KG itself [18, 37, 94] can learn from scratch with prior knowledge and work as single monolithic reinforcement learning agents. Besides the single monolithic model of reinforcement learning, other models in Table 2 are mixtures or combinations of reinforcement learning with representation learning (embeddding) and supervised learning. For example, Rel4KC [124] and RHL [109] combine different learning mechanisms, including supervised learning, to generate contrastive labeling [124] with hierarchical multi-level reinforcement learning at different levels of embedding representation [109]. Since the process of learning and KG refinement are seen as interacting agents, reinforcement learning also has the potential to combine different learning mechanisms in multiple agents [59, 136].

The issue of KG embedding representation in reinforcement learning is that the changes made to KG as parts of the refinement objective to make incremental updates can be costly and difficult, especially if they must be completed in real time. Retraining the embedding space may take time, and previous learned data may need to be discarded, but with GNN based embedding as an exception. GRL [114] makes use of GNN embedding representation (GCN) to operate on the graph. Since GNN, to an extent, supports incremental learning through iterative aggregation of neighboring entities in the graph, learned actions to generate subgraphs can be made possible in GRL. Combined with GAN to generate new subgraph and fix incorrect nodes, GRL can be operated incrementally to refine a KG. However, this still requires precise tuning of hyperparameters to allow optimal changes to the graph that conform with the particular tasks of the refinement operations.

Following the characteristics of the approaches as laid out in Tables 1 and 2, each KG refinement functionality and operation is characterized based on the machine learning process applied. It also indicates that link prediction in completing missing relations is the most common operation in all variations of machine learning including representation learning.

4.3.4 Insights and Considerations. As observed in this survey, the kind of machine learning most applied and studied in KG refinement is representation learning. This is reasonable as KGs need to be encoded in a certain way to reveal the necessary features and properties enabling most of the refinement operations. As such, representation learning can be considered as the most practical to realize as it minimizes the requirement for external data set or user-provided labels except the initially constructed KG to refine itself. Semi-supervised, self-supervised, and contrastive learning techniques may reduce the inclusion of user-defined labeling and the size of the training set while improving the prediction performance. Among different operations in KG Refinement, *completing missing relations* is the most straightforward to realize with representation learning. Other operations such as *instance matching*, *correcting errors*, and *completing types and attributes* are also supported but most require enhanced encoding techniques and combination with other machine learning models to deal with the training complexity and graph heterogeneity.

On the other hand, as discussed previously, most of the embedding models and representation learning are only effective if conducted once due to the complexity of the representation learning to learn the *encoding-decoding* model. Despite the inclusion of temporal information for dynamic representation in KGs and the incremental capability in certain types of GNN, the cost of frequent updates and *fine-tunings* makes the representation learning impractical. In this case, combining different methods of learning and the incorporation of agency models, such as those in reinforcement learning, to enable continuous updates on the graph representation can be the way for online refinement in KGs. The combination of different methods and agency in refining the KG may also be required to make the entire system complete since the encoding KG operation only produces representation for identifying potential errors, duplication, or missing information. However, they still do not include the steps to correct the KG itself. As such, learning how to update and fix the KG effectively is still a gap in KG refinement given that the current works mostly focus on representation learning.

Among all the main operations in KG refinement, *correcting errors* operation may have the least benefits from incorporating machine learning. Although some contemporary machine learning models, such as representation learning [41] and reinforcement learning [114, 124], have been applied to detect incorrect relationships, the kinds of errors detected by the learning systems are still limited to anomalous patterns in learned statistical distribution of the encoded graph [114]. Some of them still rely on external source of data explicitly providing positive and negative labels indicating their correctness [124]. Classical models of machine learning including supervised [23] and unsupervised [86] learning also rely on probabilistic or statistical model to infer the correctness. However, detecting errors in a more generic way as in PROSPERA [72], requires powerful reasoning capabilities to process high-order logical entailment and generate plausible axioms from ontologies to infer that a concept is logically correct.

5 OPEN CHALLENGES AND FUTURE STUDY

It can be shown from the review of machine learning methods above that different stages of KG refinement make use of different techniques and algorithms for learning. Current approaches to KG refinement are dominated by embedding techniques. However, the kind of operations addressed by most current refinement models converges with the completion tasks of KGs, such as relation prediction. As shown in Table 1, there are few “modern” techniques, specifically embeddings or deep learning methods, that are used to address the issues in KG instance matching, error detection, and type-attributes completion. As discussed in the previous section, despite the versatility of embedding methods, modern approaches are mainly used in certain operations, in particular link prediction and node classification. This is due to the limitation of the embedding representation in allowing incremental changes or real-time updates since re-training the embedding representation is expensive.

Accordingly, the pipeline of learning involving embedding representation can be computationally heavy [136]. For example, during every step of action selection in reinforcement learning, every time new triples are added or updated, the corresponding embedding representation of the graph should also change via re-training on the updated graph. Although GNN can be potentially leverage to resolve this issue, careful and finer-grained tuning on the hyperparameters of GNN are still required to enable it to address the incremental learning requirement.

GNN embeddings may be the ultimate technique for representation learning in this case. On the one hand, they focus on the context-dependent embeddings of every node in the graph. On the other hand, they may also provide support to cover the entire KG when the number of iterations during the aggregation of embeddings is enough. However, aggregating information from neighboring entities and parts of the KG can be a *double-edged sword*. When node features are aggregated with the structural information from neighboring entities, the specific information may be lost as the similarity distances or values regarding the structure or the semantic of the nodes become indistinguishable. This kind of issue has been recognized in LINE [104] wherein two levels of embedding should be represented separately as first-order and second-order similarity measures. In this case, one needs to treat the similarity of the adjacency structure in the graph (e.g., two nodes are currently closely linked together) and the similarity of the structural patterns separately (e.g., two nodes are similar to each other since they have the same number of outgoing relations to the same type of nodes). The main issue here is how to come up with an embedding representation such as GNN that handles these different kinds of structural similarity separately.

The real-time updating of embedding representation in reinforcement learning and GNN models has also been studied in relation to temporal KG representation or those with diachronic embeddings. For example, TIE [121] has been the ultimate model for embedding temporal dynamic KGs. Facts from new time steps can be incorporated while knowledge derived from previous ones can still be preserved. In order to handle the dynamics in a temporal KG, an experience replay buffer is used to maintain updated relations in KG and to avoid catastrophic forgetting when the incoming information is generalized and the important information from a specific input in the previous step is lost. TIE uses the experience replay to keep track the frequencies of changes and select only the changes in the current time for training. However, important and relevant updates may come not just in the latest change or at the current time but also may appear at multiple time points. Determining what needs to be maintained in memory and what to remove requires a smarter way to manage the memory. The experience replays may not just be sequential updates in time, but may also be subgraphs themselves. Dealing with more elaborated structures to handle momentary reasoning can also be an issue to study. The open challenges to be studied in the future can also be summarized as follows:

- *Real-time incremental updates on knowledge graph embedding representation.* The issue and challenge here is allowing an embedding representation to be incrementally updated. This capability may involve integrating different approaches of learning, for example through reinforcement learning and continuous aggregation in GNN embedding methods, to update the KG representation continuously. Currently, this issue is still an open challenge, though a few approaches have tackled it to some extent [114, 121].

- *Multiple Semantic-Structural Aspect in a single embedding for the entire graph.* The challenge to address here is how to come up with an embedding representation of the KG that does not just aggregate or accumulate information from neighboring entities in the entire graph, but also separates the aspect of similarity in terms of semantic and structural information. This may require an embedding model that covers information about the graph more comprehensively, but is also flexible enough to discern individual semantic and structural aspects of similarity separately.
- *Maintaining consistency and relevance in long-term continuous knowledge graph refinement.* The challenge to address here is how to maintain the consistency and relevance of the items in the KG given the dynamic nature of the incoming change of facts and reasoning process conducted to process the KG. The experience replay buffer may need to be arranged hierarchically so that multiple versions of changes of subgraphs can still be maintained without overwhelming the entire structure of the KG.

6 CONCLUSION

This survey provides a comprehensive review of various approaches to KG refinement supported by machine learning techniques. The review covers both classical models of machine learning that operate directly on the KG representation and modern techniques that leverage KG embeddings and representation learning for various operations and tasks in KG refinement. The main types of machine learning that are relevant to the refinement tasks and operations are identified. A classification of the classical and modern techniques with embedding methods has also been presented.

As a contribution of this survey, a framework has been developed for characterizing the process and operations involved in KG refinement. The classification of operations in the refinement process includes encoding KGs, correcting errors, completing types and attributes, instance matching, and completing missing relations. The framework has also been used to characterize machine learning techniques to support KG refinement. Based on this characterization, it has been revealed that representation learning that generates embedding representation of a KG is versatile for certain types of operations, such as completing missing relations, instance matching, and correcting errors as long as they do not involve extensive updates to the KG. In particular, the embedding representation from representation learning can reveal both semantics and structural relationships among nodes and relations in the KG, which is useful in matching and classifying important elements of the KG. However, classical non-embedding machine learning techniques can still be useful when semantic and structural aspects of similarity in the graph must be dealt with separately. The modern techniques of graph learning may also need a complementary process with the classical learning approaches when significant and frequent updates to a KG are required to avoid costly re-training of the embedding representation.

Through the survey, some basic issues and open challenges have also been identified. The first kind of challenge is the incremental nature of encoding KGs, which are important for processing new relationships among entities. Rapid or frequent changes to the embedding representation may still be needed in order to deal with changes required, as in online reinforcement learning. The second type of open issue is the aspect separation of structural and semantic similarity to handle the specific contexts of similarity measures in distinguishing different entities in the graphs. The last type of issue is about maintaining the consistency and relevance of the KGs to support continuous reasoning and learning in the long run.

REFERENCES

[1] Bilal Abu-Salih. 2021. Domain-specific knowledge graphs: A survey. *Journal of Network and Computer Applications* 185 (2021), 103076. DOI: https://doi.org/10.1016/j.jnca.2021.103076

[2] Tareq Al-Moslmi, Marc Gallofré Ocaña, Andreas L. Opdahl, and Csaba Veres. 2020. Named entity extraction for knowledge graphs: A literature overview. *IEEE Access* 8 (2020), 32862–32881. DOI: https://doi.org/10.1109/ACCESS.2020.2973928

[3] Samur Araújo, Duc Tran, Arjen DeVries, Jan Hidders, and Daniel Schwabe. 2012. SERIMI: Class-based disambiguation for effective instance matching over heterogeneous web data. In *Proceedings of the 15th International Workshop on the Web and Databases 2012*, Zachary G. Ives and Yannis Velegrakis (Eds.). 25–30. Retrieved from http://db.disi.unitn.eu/pages/WebDB2012/papers/p4.pdf

[4] Siddhant Arora. 2020. A survey on graph neural networks for knowledge graph completion. arXiv:2007.12374. Retrieved from http://arxiv.org/abs/2007.12374

[5] Caleb Belth, Xinyi Zheng, Jilles Vreeken, and Danai Koutra. 2020. What is normal, what is strange, and what is missing in a knowledge graph: Unified characterization via inductive summarization. *Proceedings of the Web Conference (WWW'20)*, 1115–1126. DOI: https://doi.org/10.1145/3366423.3380189

[6] Y. Bengio, A. Courville, and P. Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (aug 2013), 1798–1828. DOI: https://doi.org/10.1109/tpami.2013.50

[7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3 (2003), 1137–1155.

[8] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Euijong Whang, and Jennifer Widom. 2009. Swoosh: A generic approach to entity resolution. *The VLDB Journal* 18, 1 (jan 2009), 255–276. DOI: https://doi.org/10.1007/s00778-008-0098-x

[9] Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*. Association for Computing Machinery, New York, NY, USA, 39–48. DOI: https://doi.org/10.1145/956750.956759

[10] Christian Bizer. 2009. The emerging web of linked data. *IEEE Intelligent Systems* 24, 5 (2009), 87–92. DOI: https://doi.org/10.1109/MIS.2009.102

[11] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2013. A semantic matching energy function for learning with multi-relational data. *Machine Learning* 94, 2 (may 2013), 233–259. DOI: https://doi.org/10.1007/s10994-013-5363-6

[12] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* . Curran Associates Inc., Red Hook, NY, USA, 2787–2795.

[13] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations 2014*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6203

[14] Volha Bryl and Christian Bizer. 2014. Learning conflict resolution strategies for cross-language wikipedia data fusion. In *Proceedings of the 23rd International Conference on World Wide Web* . Association for Computing Machinery, New York, NY, USA, 1129–1134. DOI: https://doi.org/10.1145/2567948.2578999

[15] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence* . AAAI Press, 1306–1313.

[16] Haochen Chen, Syed Fahad Sultan, Yingtao Tian, Muhao Chen, and Steven Skiena. 2019. Fast and accurate network embeddings via very sparse random projection. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* . Association for Computing Machinery, New York, NY, USA, 399–408. DOI: https://doi.org/10.1145/3357384.3357879

[17] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. 2020. Knowledge graph completion: A review. *IEEE Access* 8 (2020), 192435–192456. DOI: https://doi.org/10.1109/access.2020.3030076

[18] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akhsay Krishnamurthy, and Alexander J. Smola. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning.. In *Proceedings of the 6th International Conference on Learning Representations*. 18.

[19] Rajarshi Das, Ameya Godbole, Nicholas Monath, Manzil Zaheer, and Andrew McCallum. 2020. Probabilistic case-based reasoning for open-world knowledge graph completion. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 4752–4765. DOI: https://doi.org/10.18653/v1/2020.findings-emnlp.427

[20] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. HyTE: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2001–2011. DOI: https://doi.org/10.18653/v1/D18-1225

[21] Kathrin Dentler, Ronald Cornet, Annette ten Teije, and Nicolette de Keizer. 2011. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web* 2, 2 (2011), 71–87. DOI: https://doi.org/10.3233/SW-2011-0034

[22] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence and 30th Innovative Applications of Artificial Intelligence Conference and 8th AAAI Symposium on Educational Advances in Artificial Intelligence* . AAAI Press, Article 221, 8 pages.

[23] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* . Association for Computing Machinery, New York, NY, USA, 601–610. DOI: https://doi.org/10.1145/2623330.2623623

[24] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. Metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* . Association for Computing Machinery, New York, NY, USA, 135–144. DOI: https://doi.org/10.1145/3097983.3098036

[25] Arnab Dutta, Christian Meilicke, and Simone Paolo Ponzetto. 2014. A probabilistic approach for integrating heterogeneous knowledge sources. In *Proceedings of the Semantic Web: Trends and Challenges.* Springer International Publishing, Cham, 286–301. DOI: https://doi.org/10.1007/978-3-319-07443-6_20

[26] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment* 11, 11 (Jul 2018), 1454–1467. DOI: https://doi.org/10.14778/3236187.3236198

[27] Alfio Ferrara, Andriy Nikolov, and François Scharffe. 2011. Data linking for the semantic web. *International Journal on Semantic Web and Information Systems* 7, 3 (jul 2011), 46–76. DOI: https://doi.org/10.4018/jswis.2011070103

[28] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International Conference on World Wide Web* . Association for Computing Machinery, New York, NY, USA, 413–422. DOI: https://doi.org/10.1145/2488388.2488425

[29] Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti, Francesco Draicchio, Alberto Musetti, and Paolo Ciancarini. 2012. Automatic typing of DBpedia entities. In *Proceedings of the 11th International Conference on The Semantic Web - Volume Part I* . Springer-Verlag, Berlin, 65–81. DOI: https://doi.org/10.1007/978-3-642-35176-1_5

[30] Lise Getoor and Ashwin Machanavajjhala. 2012. Entity resolution: Theory, practice, and open challenges. *Proceedings of the VLDB Endowment* 5, 12 (aug 2012), 2018–2019. DOI: https://doi.org/10.14778/2367502.2367564

[31] Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (Apr. 2020), 3988–3995. DOI: https://doi.org/10.1609/aaai.v34i04.5815

[32] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* . Association for Computing Machinery, New York, NY, USA, 855–864. DOI: https://doi.org/10.1145/2939672.2939754

[33] Lingbing Guo, Qingheng Zhang, Weiyi Ge, Wei Hu, and Yuzhong Qu. 2019. DSKG: A deep sequential model for knowledge graph completion. In *Knowledge Graph and Semantic Computing. Knowledge Computing and Language Understanding*, Jun Zhao, Frank van Harmelen, Jie Tang, Xianpei Han, Quan Wang, and Xianyong Li (Eds.). Springer, 65–77.

[34] William L. Hamilton. 2020. Background and traditional approaches. In *Graph Representation Learning.* Springer International Publishing, Cham, 9–27. DOI: https://doi.org/10.1007/978-3-031-01588-5_2

[35] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* . Curran Associates Inc., Red Hook, NY, USA, 1025–1035.

[36] Arman Hasanzadeh, Ehsan Hajiramezanali, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Semi-implicit graph variational auto-encoders. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems.* Article 961, 10712–10723 pages. Retrieved from https://proceedings.neurips.cc/paper/2019/file/fd4771e85e1f916f239624486bff502d-Paper.pdf

[37] Zhongni Hou, Xiaolong Jin, Zixuan Li, and Long Bai. 2021. Rule-aware reinforcement learning for knowledge graph reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021.* Association for Computational Linguistics, Stroudsburg, PA, USA, 4687–4692. DOI: https://doi.org/10.18653/v1/2021.findings-acl.412

[38] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020* . Association for Computing Machinery, New York, NY, USA, 2704–2710. DOI: https://doi.org/10.1145/3366423.3380027

[39] Farah Humayun, Daniel Domingo-Fernández, Ajay Abisheck Paul George, Marie Thérèse Hopp, Benjamin F. Syllwasschy, Milena S. Detzel, Charles Tapley Hoyt, Martin Hofmann-Apitius, and Diana Imhof. 2020. A computational approach for mapping heme biology in the context of hemolytic disorders. *Frontiers in Bioengineering and Biotechnology* 8, 74 (2020). DOI: https://doi.org/10.3389/fbioe.2020.00074

[40] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference* 1 (2015), 687–696. DOI: https://doi.org/10.3115/v1/p15-1067

[41] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. *Proceedings of the AAAI Conference on Artificial Intelligence* 30, 1 (Feb. 2016), 985–991. DOI: https://doi.org/10.1609/aaai.v30i1.10089

[42] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems* 33, 2 (2022), 494–514. DOI: https://doi.org/10.1109/TNNLS.2021.3070843

[43] Shengbin Jia, Yang Xiang, Xiaojun Chen, Kun Wang, and Shijia. 2019. Triple trustworthiness measurement for knowledge graph. In *Proceedings of the World Wide Web Conference* . Association for Computing Machinery, New York, NY, USA, 2865–2871. DOI: https://doi.org/10.1145/3308558.3313586

[44] Mayank Kejriwal. 2019. *Domain-specific Knowledge Graph Construction.* Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-12375-8

[45] M. Kejriwal and D. P. Miranker. 2013. An unsupervised algorithm for learning blocking schemes. In *Proceedings of the 2013 IEEE International Conference on Data Mining* . IEEE Computer Society, 340–349. DOI: https://doi.org/10.1109/ICDM.2013.60

[46] Mayank Kejriwal and Daniel P. Miranker. 2015. Semi-supervised instance matching using boosted classifiers. In *Proceedings of the Semantic Web. Latest Advances and New Domains*, Fabien Gandon, Marta Sabou, Harald Sack, Claudia d'Amato, Philippe Cudré-Mauroux, and Antoine Zimmermann (Eds.). Springer International Publishing, Cham, 388–402.

[47] Mayank Kejriwal and Daniel P. Miranker. 2015. An unsupervised instance matcher for schema-free RDF data. *Web Semantics* 35, P2 (2015), 102–123. DOI: https://doi.org/10.1016/j.websem.2015.07.002

[48] Mayank A. Kejriwal, Knoblock Craig, and Pedro Szekely. 2021. *Knowledge Graphs: Fundamentals, Techniques, and Applications.* MIT Press, Cambridge.

[49] Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. 2020. Multi-task learning for knowledge graph completion with pre-trained language models. In *Proceedings of the 28th International Conference on Computational Linguistics*, Donia Scott, Nuria Bel, and Chengqing Zong (Eds.). International Committee on Computational Linguistics, Barcelona, Spain (Online), 1737–1743. DOI: https://doi.org/10.18653/v1/2020.coling-main.153

[50] Thomas Kipf and Max Welling. 2016. Variational graph auto-encoders. arXiv: 1611.07308. Retrieved from https://arxiv.org/abs/1611.07308

[51] Jakub Klímek, Petr Škoda, and Martin Nečaský. 2019. Survey of tools for linked data consumption. *Semantic Web* 10, 4 (2019), 665–720. DOI: https://doi.org/10.3233/SW-180316

[52] Christian Kolthoff and Arnab Dutta. 2015. Semantic relation composition in large scale knowledge bases. In *Proceedings of the 3rd International Workshop on Linked Data for Information Extraction.* Gentile, Anna Lisa, Aachen, 34–47.

[53] Nihel Kooli, Robin Allesiardo, and Erwan Pigneul. 2018. Deep learning based approach for entity resolution in databases. In *Intelligent Information and Database Systems.* Springer International Publishing, Cham, 3–12. DOI: https://doi.org/10.1007/978-3-319-75420-8_1

[54] Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. In *Proceedings of the 8th International Conference on Learning Representations.* OpenReview.net. Retrieved from https://openreview.net/forum?id=rke2P1BFwS

[55] Dustin Lange, Christoph Böhm, and Felix Naumann. 2010. Extracting structured information from Wikipedia articles to populate infoboxes. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* - . ACM Press, 1661–1664. DOI: https://doi.org/10.1145/1871437.1871698

[56] Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the Web Conference 2018* . International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1771–1776. DOI: https://doi.org/10.1145/3184558.3191639

[57] Kai Lei, Jin Zhang, Yuexiang Xie, Desi Wen, Daoyuan Chen, Min Yang, and Ying Shen. 2019. Path-based reasoning with constrained type attention for knowledge graph completion. *Neural Computing and Applications* 32, 11 (apr 2019), 6957–6966. DOI: https://doi.org/10.1007/s00521-019-04181-1

[58] Huiying Li, Yuanyuan Li, Feifei Xu, and Xinyu Zhong. 2015. Probabilistic error detecting in numerical linked data. In *Database and Expert Systems Applications*. Springer International Publishing, Cham, 61–75. DOI: https://doi.org/10.1007/978-3-319-22849-5_5

[59] Zixuan Li, Xiaolong Jin, Saiping Guan, Yuanzhuo Wang, and Xueqi Cheng. 2018. Path reasoning over knowledge graph: A multi-agent and reinforcement learning based method. In *Proceedings of the 2018 IEEE International Conference on Data Mining Workshops*, Vol. 2018-Novem. IEEE, 929–936. DOI: https://doi.org/10.1109/ICDMW.2018.00135

[60] Zhuotong Li, Yongli Zhao, Yajie Li, Sabidur Rahman, Xiaosong Yu, and Jie Zhang. 2020. Demonstration of fault localization in optical networks based on knowledge graph and graph neural network. In *Proceedings of the Optical Fiber Communications Conference and Exhibition* Part F174-OFC 2020 (2020), 1–3. DOI: https://doi.org/10.1364/OFC.2020.Th1F.5

[61] Peiying Lin, Yangfan Li, Wensheng Luo, Xu Zhou, Yuanyuan Zeng, Kenli Li, and Keqin Li. 2022. Personalized query techniques in graphs: A survey. *Information Sciences* 607 (2022), 961–1000. DOI: https://doi.org/10.1016/j.ins.2022.06.023

[62] Yankai Lin, Xu Han, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2018. Knowledge representation learning: A quantitative review. arXiv:1812.10901. Retrieved from http://arxiv.org/abs/1812.10901

[63] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. *Proceedings of the National Conference on Artificial Intelligence* 3, 1 (2015), 2181–2187.

[64] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* . JMLR.org, 2168–2178.

[65] Shiheng Ma, Jianhui Ding, Weijia Jia, Kun Wang, and Minyi Guo. 2017. TransT: Type-based multiple embedding representations for knowledge graph completion. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10534 LNAI (2017), 717–733. DOI: https://doi.org/10.1007/978-3-319-71249-9_43

[66] Yanfang Ma, Huan Gao, Tianxing Wu, and Guilin Qi. 2014. Learning disjointness axioms with association rule mining and its application to inconsistency detection of linked data. In *Proceedings of the Communications in Computer and Information Science*. Springer Berlin , 29–41. DOI: https://doi.org/10.1007/978-3-662-45495-4_3

[67] Khalid Mahmood Malik, Madan Krishnamurthy, Mazen Alobaidi, Maqbool Hussain, Fakhare Alam, and Ghaus Malik. 2020. Automated domain-specific healthcare knowledge graph curation framework: Subarachnoid hemorrhage as phenotype. *Expert Systems with Applications* 145 (2020), 113120. DOI: https://doi.org/10.1016/j.eswa.2019.113120

[68] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems -* . Curran Associates Inc., Red Hook, NY, USA, 3111–3119.

[69] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 -* . Association for Computational Linguistics, USA, 1003–1011.

[70] Tom M. Mitchell. 1997. *Machine Learning*. McGraw-Hill, New York.

[71] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data* . Association for Computing Machinery, New York, NY, USA, 19–34. DOI: https://doi.org/10.1145/3183713.3196926

[72] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. 2011. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining* . Association for Computing Machinery, New York, NY, USA, 227–236. DOI: https://doi.org/10.1145/1935826.1935869

[73] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. arXiv: 1707.05005. Retrieved from https://arxiv.org/abs/1707.05005

[74] Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. 2016. A survey of current Link Discovery frameworks. *Semantic Web* 8, 3 (2016), 419–436. DOI: https://doi.org/10.3233/SW-150210

[75] Dat Quoc Nguyen. 2020. A survey of embedding models of entities and relationships for knowledge graph completion. In *Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*. 1–14. DOI: https://doi.org/10.18653/v1/2020.textgraphs-1.1

[76] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning* . Omnipress, Madison, WI, USA, 809–816.

[77] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: Scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web* . Association for Computing Machinery, New York, NY, USA, 271–280. DOI: https://doi.org/10.1145/2187836.2187874

[78] Andriy Nikolov, Mathieu d'Aquin, and Enrico Motta. 2012. Unsupervised learning of link discovery configuration. In *Proceedings of the 9th Extended Semantic Web Conference* . Springer Berlin 119–133. DOI:https://doi.org/10.1007/978-3-642-30284-8_15

[79] Andrea Giovanni Nuzzolese, Aldo Gangemi, Valentina Presutti, and Paolo Ciancarini. 2012. Type inference through the analysis of Wikipedia links. In *Proceedings of the World Wide Web 2012 Workshop on Linked Data on the Web.* 1–9. Retrieved from http://ceur-ws.org/Vol-937/

[80] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. 2021. An embedding-based approach to rule learning in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering* 33, 4 (2021), 1348–1359.

[81] A. Paccanaro and G.E. Hinton. 2001. Learning distributed representations of concepts using linear relational embedding. *IEEE Transactions on Knowledge and Data Engineering* 13, 2 (2001), 232–244. DOI:https://doi.org/10.1109/69.917563

[82] Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. 2013. Automatic expansion of DBpedia exploiting wikipedia cross-language information. In *Proceedings of the Semantic Web: Semantics and Big Data*, Philipp Cimiano, Oscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph (Eds.). Springer Berlin 397–411.

[83] Heiko Paulheim. 2012. Browsing linked open data with auto complete. In *Proceedings of the 10th Semantic Web Challenge 2012 at the 11th International Semantic Web Conference.* 1–8.

[84] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web* 8, 3 (2017), 489–508.

[85] Heiko Paulheim and Christian Bizer. 2013. Type inference on noisy RDF data. In *Proceedings of the 12th International Semantic Web Conference - Part I.* Springer Berlin 510–525. DOI:https://doi.org/10.1007/978-3-642-41335-3_32

[86] Heiko Paulheim and Christian Bizer. 2014. Improving the quality of linked data using statistical distributions. *International Journal on Semantic Web and Information Systems* 10, 2 (apr 2014), 63–86. DOI:https://doi.org/10.4018/ijswis.2014040104

[87] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* . Association for Computing Machinery, New York, NY, 701–710. DOI:https://doi.org/10.1145/2623330.2623732

[88] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. 2013. Knowledge graph identification. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8218 LNCS, PART 1 (2013), 542–557. DOI:https://doi.org/10.1007/978-3-642-41335-3_34

[89] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and Node2vec. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining* . Association for Computing Machinery, New York, NY, USA, 459–467. DOI:https://doi.org/10.1145/3159652.3159706

[90] Stuart J. Russell and Peter Norvig. 2020. *Artificial Intelligence: A Modern Approach (4th Edition).* Pearson, Hoboken. Retrieved from http://aima.cs.berkeley.edu/

[91] David Schindler, Benjamin Zapilko, and Frank Krüger. 2020. Investigating software usage in the social sciences: A knowledge graph approach. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12123 LNCS (2020), 271–286. DOI:https://doi.org/10.1007/978-3-030-49461-2_16 arXiv:2003.10715. Retrieved from https://arxiv.org/abs/2003.10715

[92] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the 15th European Semantic Web Conference.* Springer International Publishing, 593–607. DOI:https://doi.org/10.1007/978-3-319-93417-4_38

[93] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (jul 2019), 3060–3067. DOI:https://doi.org/10.1609/aaai.v33i01.33013060

[94] Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. M-Walk: Learning to walk over graphs using Monte Carlo tree search. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* . Curran Associates Inc., Red Hook, NY, USA, 6787–6798.

[95] Jiawei Sheng, Shu Guo, Zhenyu Chen, Juwei Yue, Lihong Wang, Tingwen Liu, and Hongbo Xu. 2020. Adaptive attentional network for few-shot knowledge graph completion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing.* 1681–1691. DOI:https://doi.org/10.18653/v1/2020.emnlp-main.131

[96] Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence* , 1957–1964.

[97] Jennifer Sleeman and Tim Finin. 2013. Type prediction for efficient coreference resolution in heterogeneous semantic graphs. In *Proceedings of the 2013 IEEE 7th International Conference on Semantic Computing.* 78–85. DOI:https://doi.org/10.1109/ICSC.2013.22

[98] Jennifer Sleeman, Tim Finin, and Anupam Joshi. 2015. Topic modeling for RDF graphs. In *Proceedings of the 3rd International Workshop on Linked Data for Information Extraction.* CEUR Workshop Proceedings, 48–62.

[99] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1* . Curran Associates Inc., Red Hook, NY, USA, 926–934.

[100] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1* . Curran Associates Inc., Red Hook, NY, USA, 926–934.

[101] Tommaso Soru and Axel-Cyrille Ngonga Ngomo. 2014. A comparison of supervised learning classifiers for link discovery. In *Proceedings of the 10th International Conference on Semantic Systems* . Association for Computing Machinery, New York, NY, USA, 41–44. DOI: https://doi.org/10.1145/2660517.2660532

[102] Zhiqing Sun, Zhi Hong Deng, Jian Yun Nie, and Jian Tang. 2019. RotatE: Knowledge graph embedding by relational rotation in complex space. In *Proceedings of the 7th International Conference on Learning Representations* .

[103] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction.* A Bradford Book, Cambridge, MA, USA.

[104] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web* . International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1067–1077. DOI: https://doi.org/10.1145/2736277.2741093

[105] Anqi Tian, Chunhong Zhang, Miao Rang, Xueying Yang, and Zhiqiang Zhan. 2020. RA-GCN: Relational aggregation graph convolutional network for knowledge graph completion. In *Proceedings of the 2020 12th International Conference on Machine Learning and Computing* . Association for Computing Machinery, New York, NY, USA, 580–586. DOI: https://doi.org/10.1145/3383972.3384067

[106] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-Evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* . JMLR.org, 3462–3471.

[107] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.), Vol. 48. PMLR, 2071–2080. Retrieved from https://proceedings.mlr.press/v48/trouillon16.html

[108] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR '18).* Retrieved from https://openreview.net/forum?id=rJXMpikCZ

[109] Guojia Wan, Shirui Pan, Chen Gong, Chuan Zhou, and Gholamreza Haffari. 2020. Reasoning like human: Hierarchical reinforcement learning for knowledge graph reasoning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI-20*, Christian Bessiere (Ed.). International Joint Conferences on Artificial Intelligence Organization, 1926–1932. DOI: https://doi.org/10.24963/ijcai.2020/267 Main track.

[110] Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021. Structure-augmented text representation learning for efficient knowledge graph completion. In *Proceedings of the Web Conference 2021 - Proceedings of the World Wide Web Conference, WWW 2021*, 1737–1748. DOI: https://doi.org/10.1145/3442381.3450043 arXiv:2004.14781

[111] Bin Wang, Guangtao Wang, Jing Huang, Jiaxuan You, Jure Leskovec, and C.-C. Jay Kuo. 2021. Inductive learning on commonsense knowledge graph completion. In *Proceedings of the 2021 International Joint Conference on Neural Networks.* 1–8. DOI: https://doi.org/10.1109/ijcnn52387.2021.9534355

[112] Changping Wang, Chaokun Wang, Zheng Wang, Xiaojun Ye, and Philip S. Yu. 2020. Edge2vec. *ACM Transactions on Knowledge Discovery from Data* 14, 4 (jul 2020), 1–24. DOI: https://doi.org/10.1145/3391298

[113] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* . Association for Computing Machinery, New York, NY, USA, 1225–1234. DOI: https://doi.org/10.1145/2939672.2939753

[114] Qi Wang, Yuede Ji, Yongsheng Hao, and Jie Cao. 2020. GRL: Knowledge graph completion with GAN-based reinforcement learning. *Knowledge-Based Systems* 209 (2020), 106421. DOI: https://doi.org/10.1016/j.knosys.2020.106421

[115] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743. DOI: https://doi.org/10.1109/TKDE.2017.2754499

[116] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the National Conference on Artificial Intelligence* , 1112–1119. Retrieved from https://wvvw.aaai.org/ojs/index.php/AAAI/article/view/4725

[117] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3-4 (may 1992), 279–292. DOI: https://doi.org/10.1007/BF00992698

[118] Yuze Wei, Jie Luo, and Huiyuan Xie. 2016. KGRL: An OWL2 RL reasoning system for large scale knowledge graph. In *Proceedings of the 2016 12th International Conference on Semantics, Knowledge and Grids*. 83–89. DOI: https://doi.org/10.1109/SKG.2016.020

[119] Dominik Wienand and Heiko Paulheim. 2014. Detecting incorrect numerical data in DBpedia. In *Proceedings of the 11th European Semantic Web Conference*. Springer International Publishing, 504–518. DOI: https://doi.org/10.1007/978-3-319-07443-6_34

[120] Pornpit Wongthongtham and Bilal Abu Salih. 2018. Ontology-based approach for identifying the credibility domain in social Big Data. *Journal of Organizational Computing and Electronic Commerce* 28, 4 (2018), 354–377. DOI: https://doi.org/10.1080/10919392.2018.1517481

[121] Jiapeng Wu, Yishi Xu, Yingxue Zhang, Chen Ma, Mark Coates, and Jackie Chi Kit Cheung. 2021. TIE: A framework for embedding-based incremental temporal knowledge graph completion. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* . Association for Computing Machinery, New York, NY, USA, 428–437. DOI: https://doi.org/10.1145/3404835.3462961

[122] Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. 2021. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence* 2, 2 (2021), 109–127. DOI: https://doi.org/10.1109/tai.2021.3076021. arXiv:2105.00696

[123] Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. TransG: A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers* 4 (2016), 2316–2325. DOI: https://doi.org/10.18653/v1/p16-1219

[124] Xiao Lin, Pero Subasic, and Hongfeng Yin. 2020. Rel4KC: A reinforcement learning agent for knowledge graph completion and validation. In *Proceedings of the Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 1291. Retrieved from http://www.cse.msu.edu/$\sim$zhaoxi35/DRL4KDD/1.pdf

[125] Qianqian Xie, Jimin Huang, Pan Du, Min Peng, and Jian-Yun Nie. 2021. Inductive topic variational graph autoencoder for text classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 4218–4227. DOI: https://doi.org/10.18653/v1/2021.naacl-main.333

[126] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 564–573. DOI: https://doi.org/10.18653/v1/d17-1060. arXiv:1707.06690

[127] Bingcong Xue and Lei Zou. 2023. Knowledge graph quality management: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 5 (2023), 4969–4988. DOI: https://doi.org/10.1109/TKDE.2022.3150080

[128] Yexiang Xue, Yang Yuan, Zhitian Xu, and Ashish Sabharwal. 2018. Expanding holographic embeddings for knowledge completion. In *Proceedings of the Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2018/file/dd28e50635038e9cf3a648c2dd17ad0a-Paper.pdf

[129] Hehua Yan, Jun Yang, and Jiafu Wan. 2020. KnowIME: A system to construct a knowledge graph for intelligent manufacturing equipment. *IEEE Access* 8 (2020), 41805–41813. DOI: https://doi.org/10.1109/ACCESS.2020.2977136

[130] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the 3rd International Conference on Learning Representations*, Yoshua Bengio and Yann LeCun (Eds.). Retrieved from http://arxiv.org/abs/1412.6575

[131] Cheng Yang, Zhiyuan Liu, Cunchao Tu, and Maosong Sun. 2017. Network representation learning: an overview. *SCIENTIA SINICA Informationis* 47, 8 (2017), 980–996. DOI: https://doi.org/10.1360/n112017-00145

[132] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. arXiv: 1909.03193. Retrieved from http://arxiv.org/abs/1909.03193

[133] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Network representation learning: A survey. *IEEE Transactions on Big Data* 6, 1 (2018), 3–28. DOI: https://doi.org/10.1109/tbdata.2018.2850013

[134] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. ProNE: Fast and scalable network representation learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* . AAAI Press, 4278–4284.

[135] Rui Zhang, Dimitar Hristovski, Dalton Schutte, Andrej Kastrin, Marcelo Fiszman, and Halil Kilicoglu. 2021. Drug repurposing for COVID-19 via knowledge graph completion. *Journal of Biomedical Informatics* 115 (2021), 103696. DOI: https://doi.org/10.1016/j.jbi.2021.103696

[136] Yunan Zhang, Xiang Cheng, Heting Gao, and ChengXiang Zhai. 2019. Cooperative reasoning on knowledge graph and corpus: A multi-agent reinforcement learning approach. arXiv:1912.02206. Retrieved from https://arxiv.org/abs/1912.02206

[137] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the AAAI 2020 - 34th AAAI Conference on Artificial Intelligence* (2020), 3065–3072. DOI: https://doi.org/10.1609/aaai.v34i03.5701

[138] Yu Zhao, Sheng Gao, Patrick Gallinari, and Jun Guo. 2015. Knowledge base completion by learning pairwise-interaction differentiated embeddings. *Data Mining and Knowledge Discovery* 29, 5 (jul 2015), 1486–1504. DOI: https://doi.org/10.1007/s10618-015-0430-1

[139] Xiaofei Zhou, Qiannan Zhu, Ping Liu, and Li Guo. 2017. Learning knowledge embeddings by combining limit-based scoring loss. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* . Association for Computing Machinery, New York, NY, USA, 1009–1018. DOI: https://doi.org/10.1145/3132847.3132939

[140] Xiaojin Zhu and Andrew B. Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3, 1 (jan 2009), 1–130. DOI: https://doi.org/10.2200/S00196ED1V01Y200906AIM006

[141] Yao Zhu, Hongzhi Liu, Zhonghai Wu, Yang Song, and Tao Zhang. 2020. Representation learning with ordered relation paths for knowledge graph completion. In *Proceedings of the EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference* (2020), 2662–2671. DOI: https://doi.org/10.18653/v1/d19-1268. arXiv:1909.11864

[142] Xiaohan Zou. 2020. A survey on application of knowledge graph. *Journal of Physics: Conference Series* 1487, 1 (03 2020), 012016. DOI: https://doi.org/10.1088/1742-6596/1487/1/012016

Received 20 April 2022; revised 5 September 2023; accepted 15 December 2023