Predicting Entity Relations across Different Security Databases by Using Graph Attention Network

Liu Yuan[1], Yude Bai[1], Zhenchang Xing[2], Sen Chen[1]*, Xiaohong Li[1]*, Zhidong Deng[3]
[1]*College of Intelligence and Computing, Tianjin University, Tianjin, China*
[2]*Research School of Computer Science, Australian National University, Data61 CSIRO, Australia*
[3]*State Grid Customer Service Center, Tianjin, China*
{*yuanliuyl, baiyude, senchen, xiaohongli*}*@tju.edu.cn, zhenchang.xing@anu.edu.au*

***Abstract*—Security databases such as Common Vulnerabilities and Exposures (CVE), Common Weakness Enumeration (CWE), and Common Attack Pattern Enumeration and Classification (CAPEC) maintain diverse high-quality security concepts, which are treated as security entities. Meanwhile, security entities are documented with many potential relation types that profit for security analysis and comprehension across these three popular databases. To support reasoning security entity relationships, translation-based knowledge graph representation learning treats each triple independently for the entity prediction. However, it neglects the important semantic information about the neighbor entities around the triples. To address it, we propose a text-enhanced graph attention network model (text-enhanced GAT). This model highlights the importance of the knowledge in the $2-$hop neighbors surrounding a triple, under the observation of the diversity of each entity. Thus, we can capture more structural and textual information from the knowledge graph about the security databases. Extensive experiments are designed to evaluate the effectiveness of our proposed model on the prediction of security entity relationships. Moreover, the experimental results outperform the state-of-the-art by Mean Reciprocal Rank (MRR) 0.132 for detecting the missing relationships.**

***Index Terms*—Security entity, Entity relation prediction, Security database, Knowledge graph, Graph attention network**

I. Introduction

Software with vulnerabilities and weaknesses when coding or implementing are compromised more easily by attackers through various attack patterns [1]–[4]. Security experts maintain a series of security databases to manage the document of software vulnerabilities, weaknesses, and attack patterns, in order to protect security and spread security knowledge. Common Vulnerabilities and Exposures (CVE) [5] is a list of publicly identified security vulnerabilities. For example, *CVE-2017-0327* is an elevation of privilege vulnerability in the NVIDIA crypto driver. The Common Weakness Enumeration (CWE) [6] presents software weaknesses developed by the security community such as *CWE-120* is a classic buffer overflow. Common Attack Pattern Enumeration and Classification (CAPEC) [7] defines specific attack patterns and relevant solutions to defense. The *CAPEC-10* denotes its attack pattern by "Buffer Overflow via Environment Variables" as an example.

As entities in the three security databases, the CVEs, CWEs, and CAPECs create a large number of relations (see in Fig. 1).

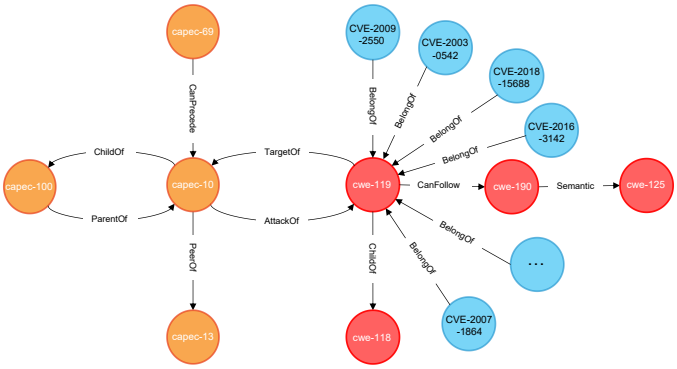

Fig. 1. An illustration of security knowledge graph across CVEs, CWEs and CAPECs.

For instance, the *CVE-2018-15688* is a specific "Overflow" vulnerability of *CWE-119* via the relation *BelongOf*, and the *CAPEC-100* is the parent of *CAPEC-10*, which highly summarize the attack pattern of buffer overflow. Obviously, security entity relations give plenty of security information that benefits experts for security analysis and further vulnerability repair [8]. However, we observe the fact of missing relations, such as the *CAPEC-111* (JSON Hijacking (aka JavaScript Hijacking)) is the "child" of the *CAPEC-116* (Data Excavation Attacks) that the relation *ChildOf* is appended until the CWE database version 1.5. The literature [9] also clarifies the importance of predicting the security entity relations.

Knowledge graph representation learning, which has been widely used in text matching [10], path reasoning [11], and question answering [12], is proved to effectively predict the implicit security entity relations from the documentation of the security databases [8], [13]. But the translational models in [8], [9] learn structural and textual embeddings by using simple operations and few parameters which are the low-quality embeddings. Furthermore, they consider the triple independently so that the models cannot capture the potential relations in the adjacent neighbors of each entity. Inspired by the graph attention networks (GAT) [14] and the investigation of the diversity of entity relations (i.e., *CWE-119* is *TargetOf CAPEC-10* while *ChildOf CWE-118* in the graph.), we propose a text-enhanced GAT model by representing each entity with the

*Sen Chen and Xiaohong Li are the corresponding authors.

latent knowledge from its 2−hop neighbors. In this model, we firstly embed the graph structure through TransE [15] and the entity descriptions via Word2Vec [16] and convolutional neural network (CNN). After concatenating the knowledge of structure-embeddings and description-embeddings from the 2−hop neighbors of a given entity, the attention layers are followed to weigh different entities to guide the entity relations prediction task.

Extensive experiments are conducted to investigate the efficiency of our proposed text-enhanced GAT model. For the newly downloaded CVE, CWE, and CAPEC entity relations and relevant descriptions before July 20, 2020, we create a security knowledge graph including $5,416$ entities ($4,003$ CVEs, 891 CWEs, and 522 CAPECs) and $10,502$ relations (9 types). Our experimental results on this graph outperform the state-of-the-art [9] by Mean Reciprocal Rank (MRR) 0.132 of the overall performance when detecting the missing relationships across existing entities. We then explore the effectiveness of predict-head and predict-tail tasks for each relation type. Additionally, the ablation study is further implemented to inspect which part of the modules in our given model is the most important one for the link prediction.

In summary, we make the following contributions:

- We design an advanced text-enhanced GAT model to better represent and learn the structural and textual knowledge from the security knowledge graph, which integrates software vulnerabilities, weaknesses, and attack patterns.
- We consider the knowledge of 2−hop adjacent nodes as the additional information to enrich the entity relation features of a given security entity.
- We measure the performance of our proposed model through extensive experiments and clarify the superiority of the graph attention network for predicting relations of security entities.

II. Background

This section describes three security databases, i.e., Common Vulnerabilities and Exposures (CVE) [5], Common Weakness Enumeration (CWE) [6], and Common Attack Pattern Enumeration and Classification (CAPEC) [7]. All of them have internal dependency relations and external dependencies between each other, as shown in Fig. 1. Then we will also explain the necessary part related to knowledge graph representation learning.

A. Common Vulnerabilities and Exposures

As a well-known security vulnerability records, Common Vulnerabilities and Exposures database [5] provides a unique identification number (i.e., *CVE-2009-2550* in Fig. 1) for each public security vulnerability or exposure. Besides, one CVE in the database has a standard textual description and at least one public reference. It is beneficial for understanding security vulnerability and evaluating relevant services, tools, and databases [17] under such a dictionary structure, so that a more understandable CVE-related database with abundant semantic security knowledge needs to be established.

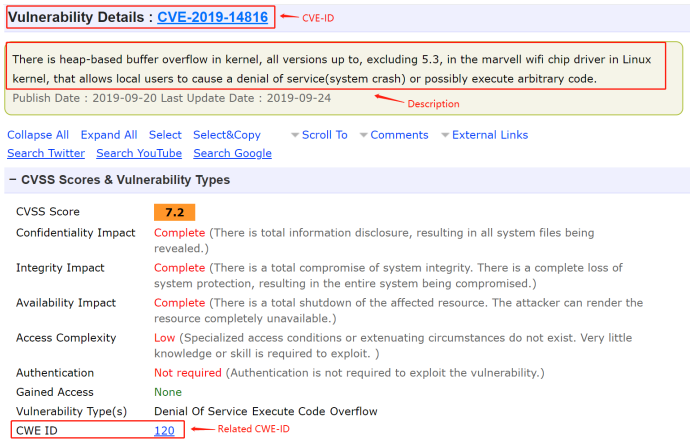

Fig. 2. An example of the vulnerability *CVE-2019-14816* report

National Vulnerability Database (NVD) [18], the U.S. government repository of standards-based vulnerability management data, supplements an additional analysis of the vulnerability, security databases, and a fine-grained search engine for the CVE database. Specifically, it includes databases of security checklist references, security software vulnerabilities, software configuration errors, product names, and impact metrics. The NVD also helps to automatically manage vulnerability, update security metrics (all updates to the CVE database appear immediately on the NVD database).

We collect and summary all information of CVEs from a well-maintained website CVE details [19] that integrates the CVE vulnerability data taken from the CVE database and the NVD database. This website also combines the relevant security modules of Exploit [20]. Thus we extract the CVE vulnerability information from the CVE details and aim to dig out more comprehensive security knowledge for assistants. For each vulnerability, we crawl the CVE-ID, textual description, and corresponding CWE-ID of this vulnerability, based on the given web interface for vulnerability reports in the webpage. Specifically, we regard that the CVE is associated with one CWE as *BelongOf*, such as the *CVE-2019-14816* is *BelongOf CWE-120* (see in Fig. 2).

B. Common Weakness Enumeration

Common Weakness Enumeration (CWE) is a list of software and hardware weakness types developed by the security community. The weakness defined by the CWE database contains vulnerabilities, bugs, flaws, faults, and other errors when developers design, code, or implement the software and hardware. The network and even the whole system with those weaknesses could be under attack [7]. The CWE database provides abundant security knowledge relevant to those weaknesses, such as measuring sticks for security tools, services for inspecting source code, and effective advice for designing safe software architecture [6].

Under the guidance of the CWE database, we obtain security recommendations on “Software Development”, “Hardware Design”, and “Research Concepts”. We focus on the “Re-

TABLE I
DISTRIBUTION STATISTICS OF CWE RELATIONSHIP

Type	BelongOf	AttackOf	ChildOf	ParentOf	CanPrecede	CanFollow	PeerOf	Semantic
Number	4,003	1,067	1,084	1,084	129	129	113	713

search Concepts", which includes both software and hardware concepts and is the most critical issue. It gives the relations between CWEs, such as *CWE-59* is *ChildOf CWE-706* (see in Table I), which help security researchers to find the potential knowledge of different CWEs.

We crawl 891 CWEs from the CWE database based on the Research Concepts (Version 4.2). Each CWE has its CWE-ID, textual description, relations of other CWEs, and related attack patterns (i.e., the specific attack methods pointed at this CWE). As is shown in Table I, we define 8 relations about the CWE databases, which the *BelongOf* is the relation between CWE and CVE, the *AttackOf* refers to the relation between CWE and CAPEC, the *ChildOf*, *ParentOf*, *CanPrecede*, *CanFollow* and *PeerOf* denote the relations between two CWEs, and the *Semantic* shows the relation between one CWE and another webpage hyperlink CWE in its weakness report.

C. *Common Attack Pattern Enumeration and Classification*

Common Attack Pattern Enumeration and Classification database [7] provides a comprehensive dictionary of known attack patterns (CAPEC) that are employed by attackers to exploit the software or hardware weaknesses in the applications and systems. It helps analysts and developers to understand how to operate attack patterns and how to advance cybersecurity. In the CAPEC database, an attack pattern includes the common attributes and approaches taken by attackers. It also presents the challenges to exploit the known weaknesses and solve the potential risk. For example, *CAPEC-34* means the attack pattern "HTTP Response Splitting", *CAPEC-66* refers to *SQL Injection*, and *CAPEC-100* denotes *Buffer Overflow*.

For each CAPEC in the CAPEC database, we consider the description of the attack pattern, relations, and the relevant weaknesses as the critical features in the experiments. Furthermore, the relation types of CAPEC are set as *ChildOf*, *ParentOf*, *CanPrecede*, *CanFollow*, and *PeerOf* that all of those present the relation between two CAPECs. And we regard the relation between CAPEC and its related weakness (CWE) as *TatgetOf*. Table II gives the detailed distribution. Note that the number of *TatgetOf* in Table II is $1,067$ which is equal to the number of *AttackOf* in Table I. Because they are just different aspects to explain the relation between weakness (CWE) and attack pattern (CAPEC).

D. *Knowledge Graph Representation Learning*

The knowledge graph representation learning attempts to represent entity and relation into a low-dimensional dense vector. Then the training model based on representation learning continuously optimize the hyperparameters in the function $f(h, r, t)$ and predict the missing element in the given triple $\langle h, r, t\rangle$ (see the specific definition in Section III-A). For instance, given any two elements of the triples <*CAPEC-54* (Query System for Information), *ChildOf*, *CAPEC-116* (Excavation)>, we can infer the other one by the training neural network model. We adopt TransE [15] to initialize neural vectors of the entity-relation structure, and adopt graph attention neural network [14], [21] to concatenate the structure information and textual information for later prediction.

TABLE II
DISTRIBUTION STATISTICS OF CAPEC RELATIONSHIP

Type	ChildOf	ParentOf	CanPrecede	CanFollow	PeerOf	TargetOf
Number	495	495	77	33	13	1,067

III. APPROACH

In this section, we present the overview of our approach as shown in Fig. 3. This encoder-decoder model architecture aims to predict entity relations in the generated security knowledge graph, based on the proposed text-enhanced graph attention neural network.

A. *Security Knowledge Graph Generation*

We first construct a security knowledge graph as shown in Fig. 3 a), based on all the security knowledge of the CVE, CWE, and CAPEC databases. We formulate this security knowledge graph as $G = (E, R, S)$, where E, R, and S represent the set of all entities, relations, and triples respectively. Each triple S is defined as $S = \langle h, r, t\rangle$, where h, r, and t denote the head entity, relation, and tail entity, respectively. For instance, in the triple <*CVE-2010-5321*, *BelongOf*, *CWE-190*> in Fig. 1, the node *CVE-2010-5321* is the head entity, the node *CWE-190* is the tail entity, and the edge *BelongOf* is the relation between them. This graph consists of $4,003$ CVEs, 891 CWEs and 522 CAPECs. As is shown in Table III, the CVE, CWE, and CAPEC databases have 9 categories of relations. We regard 6 out of 9 categories are internal relations in which both head entity and tail entity belong to the same database (i.e., CWE or CAPEC database). The other 3 categories are external relations that the head entity and tail entity come from different databases. Note that the external relation contains abundant semantic knowledge from the textual description. Fig. 4 shows an example <*CWE-120*, *TargetOf*, *CAPEC-10*> with an external relation *TargetOf*. We observe that it specifically explains the reason why the program with weaknesses may lead to a vulnerability and how the attacker exploits such vulnerability with an attack pattern.

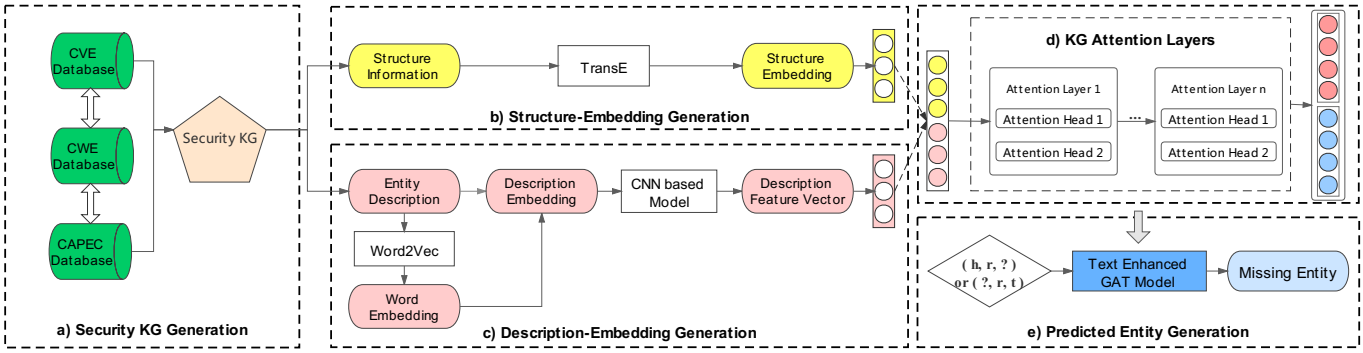

Fig. 3. Overview of our approach. KG refers to the knowledge graph.

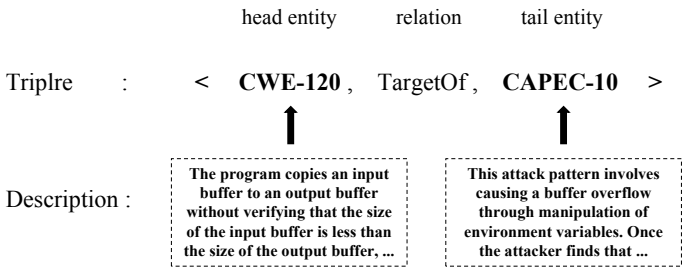

Fig. 4. A triplet example in the knowledge graph

B. *Structure-Embedding and Description-Embedding Generation*

Two strategies to embed the created security knowledge graph in Section III-A are shown as follows.

Structure-Embedding Generation. Given the structure information of entities and relations in the security knowledge graph, as shown in Fig. 3 b, we adopt TransE [15] to train such triples as the initial structure embedding. Thus we avoid the negative influence of the structure information ignored by random initialization, and efficiently obtain relational knowledge across the three security databases. We set this structure-embedding vector with a dimension of 100.

Description-Embedding Generation. As is shown in Fig. 3 c), this section aims at representing the textual description of each CVE, CWE, and CAPEC, and extracting as many semantic knowledge features as possible.

We firstly tokenize the textual sentences, remove stop words, and stem those sentences by using NLTK (a python NLP toolkit). The tokens (words) are sent into a word2vec model [16] that has been widely used in [8], [22], [23]. Each word x is transformed into a vector with the dimension of 100. We concatenate all word vectors in every entity description so that a sentence vector is produced by using the formulation $D_n = x_1 \oplus x_2 \oplus ... \oplus x_n$, where $\oplus$ means the connection operation and $x_i \in X$ (X is the word dictionary of all databases). Note that we set the sentence length as 375 (the max sentence length). If the length of one entity description is less than 375, we append 0 as substitution until its length is as long as 375. And we initialize the relevant vector as 0 with the dimension of 100 as well.

Those 375×100 sentence vectors are fed into a two-layer convolutional neural network (CNN). We set the width of the kernel as 100 to scan the convolutional feature space and apply the ReLU for activation operation. The first CNN layer is followed by a k-max ($k = 20$) pooling layer. We adopt k-max pooling instead of max pooling because the former better represents the frequent features and obtains the position information in the feature space. Then the convolutional neural feature is sent into the second CNN layer. After activated by ReLU, we use an average pooling layer to squeeze such neural features and gain its global average feature.

Finally, those structural feature vectors produced by TransE (100 dimension) and textual feature vectors produced by CNN (100 dimension) are concatenated into a feature vector with the dimension of 200.

C. *Knowledge Graph Attention Layer*

We have extracted features based on two different dimensions of structural and textual description through concatenation operation and multi-layer perceptron, and obtained a vector expression that combines two vector representations.

In our proposed security knowledge graph, we observe that the same entity plays different roles via the relations they are linked with. For example, *CAPEC-10* is *ChildOf CAPEC-100* in the graph. At the same time, it is *PeerOf CAPEC-13*. This inspires us to devote to the $2-$hop neighbors around all entities, and the reason why we use $2-$hop is that $2-$hop between entities account for the vast majority. Therefore, one entity acquires additional entity and relation knowledge from its $2-$hop neighbors. In this path of $2-$hop neighbors, the embedding values of relations are summed up which is considered as an expended relation embedding value. Meanwhile, the embedding values of entities are normalized after each graph attention layer (see in Fig. 3 d)) in order to prevent the issue of the state explosion.

In detail, we propose a multi-head attention neural network layer to extract those extending $2-$hop entity and relation features for generating a concatenated vector $\rho_{(h,r,t)}$, as shown in Fig. 3 d). Then a single-layer feedforward neural network

is followed with the inspiration of attention mechanism. We adopt the activation function LeakyReLU to calculate the attention weight $\alpha_{(h,r,t)}$ of each neighbor node of the entity h in a triple before it is normalized (Equation 1),

$$\alpha_{(h,r,t)} = LeakyReLU(\boldsymbol{W}\rho_{(h,r,t)}). \quad (1)$$

$\boldsymbol{W}$ is the linear transformation matrix. To get the normalized attention value, a *softmax* layer is applied to update the attention coefficient is $\beta_{(h,r,t)}$ through

$$\beta_{(h,r,t)} = \frac{\exp\left(\alpha_{(h,r,t)}\right)}{\sum_{n\in\mathcal{N}_h}\sum_{r'\in\mathcal{R}_{hn}}\exp\alpha_{(h,r',n)}}, \quad (2)$$

where $\mathcal{N}_h$ denotes a set of neighbor nodes of entity h, and $\mathcal{R}_{hn}$ is a set of relations between entity h and its neighbor node n. So the new embedding value of this entity is updated with the sum of all neighbor node attention values after normalization operation in Equation 2. In the next step, we perform a linear transformation on this updated embedding value of the entity. The initial embedding value of this entity is added into its linear transformed embedding value which is to avoid losing its initial embedding information during the process of learning. With such multi-head attention neural network, we learn the deeper neural features of the entity and relations.

D. Model Training

We apply hinge-loss as the loss function to optimize our proposed model, which is shown in Equation (3),

$$\mathcal{L} = \sum_{(h,r,t)\in S}\sum_{(h,r,t)'\in S'} max\{f_{(h,r,t)} - f_{(h',r,t')} + \gamma, 0\}, \quad (3)$$

where (h,r,t) represents a valid triple in the set of valid triples S and $(h,r,t)'$ refers to an invalid triple with the head entity or tail entity randomly replaced in the set of invalid triples S'. Besides, S' is given formally as

$$S' = \{(h',r,t)|h' \in E \setminus h\} \cup \{(h,r,t')|t' \in E \setminus t\}. \quad (4)$$

The $f_{(h,r,t)}$ in Equation 3 defines the L2-norm dissimilarity measure $f_{(h,r,t)} = \|h + r - t\|_{\frac{1}{2}}^2$ to be minimized for optimization. The hyperparameter γ is a margin hyper-parameter to determine the boundary of the correct and wrong triples, which is set larger than 0.

E. Decoder Design

Given the encoded entity and relation feature vector of one triple $<h,r,t>$, we apply ConvKB [24] to decode such feature vector. The convolutional layer in ConvKB is beneficial for analyzing the global embedding values of the triple when traversing each feature dimension. It also generates a translation characteristic. We use ReLU to activate the output value of the convolutional layer. Then the score function is formulated as,

$$\mathcal{F} = concat\Big(ReLU\big(\left[\vec{h},\vec{r},\vec{t}\right] * \Omega\big)\Big)\boldsymbol{W}, \quad (5)$$

where Ω denotes the convolutional filter, $\left[\vec{h},\vec{r},\vec{t}\right]$ is the vector representation of the entity and relation vector after the concatenation operation, and $*$ means the convolutional operation between them. After activated by ReLU, we concatenate multiple feature maps of the same dimension by the function of $concat()$, then we multiply them with a linear transformation matrix $\boldsymbol{W} \in R^{nk\times 1}$ to produce the final score of the triple $<h,r,t>$. Different from the encoder model, we adopt the soft-margin loss as the loss function shown in Equation (6),

$$\mathcal{L} = \sum_{(h,r,t)\in\{S\cup S'\}} \log(1 + \exp\left(l(h,r,t)\cdot\mathcal{F}\right)) + \frac{\lambda}{2}\parallel \boldsymbol{W} \parallel_2^2, \quad (6)$$

where $l(h,r,t)$ is defined as Equation (7),

$$l(h,r,t) = \begin{cases} 1 & for(h,r,t) \in S \\ -1 & for(h,r,t) \in S' \end{cases}. \quad (7)$$

In addition, we employ the batch gradient descent method to update all hyperparameters based on the adam algorithm, which helps to decrease the training time and search the global optimum value.

IV. Experiment

In this section, extensive experiments are conducted to evaluate and discuss the effectiveness of our text-enhanced GAT approach compared with the baselines, followed by three Research Questions (RQ):

- **RQ1**: How effective is our text-enhanced GAT approach in security entity relationship predictions, compared with the baseline method?
- **RQ2**: Can our approach predict well in each type of security entity relationship?
- **RQ3**: How sensitive can our approach learn from the complexity of structural and textual embedding compared with the ablation studies?

A. Experiment Setup

1) Dataset: We use BeautifulSoup[1] to crawl and parse webpages of the CVE, CWE, and CAPEC databases. For the webpages of each CVE, we collect its textual descriptions and corresponding CWE-ID. And we get CWE descriptions of each CWE and 8 relations of them such as *ChildOf*, *ParentOf*, etc. The textual descriptions of the CVE database and the CWE database only contain the systems such as Debian Linux, Linux Kernel, and Ubuntu Linux, before July 20, 2020. As attack patterns are more important in the CAPEC databases, we mainly use the information textual descriptions of each CAPEC and 6 relations of them such as *CanPrecede*, *CanFollow*, etc. We collect the newest CWE-CAPEC relations and relevant descriptions based on the CWE database with Version 4.2 and the CAPEC database with Version 3.3. To sum up, we create a security knowledge graph including 5416 entities (i.e., 4,003 CVEs, 891 CWEs and 522 CAPECs), 9 relationship types (the distribution of relations are shown in

[1]BeautifulSoup, https://beautifulsoup.readthedocs.io/zh_CN/v4.4.0/.

TABLE III
DISTRIBUTION OF RELATIONS IN THE SECURITY KNOWLEDGE GRAPH

Type	**Internal relation**						**External relation**		
	ChildOf	ParentOf	CanPrecede	CanFollow	PeerOf	Semantic	BelongOf	AttackOf	TargetOf
Number	1,579	1,579	206	162	126	713	4,003	1,067	1,067

TABLE IV
OVERALL PERFORMANCE OF LINK PREDICTION COMPARED WITH THE BASELINE

Prediction	Model	MR (Filtered)	MR (Raw)	MRR (Filtered)	MRR (Raw)	Hits@N @5(Filtered)	@5(Raw)	@10(Filtered)	@10(Raw)
Head entity $\langle ?, r, t\rangle$	TransH combined text model[1]	197	338	0.536	0.214	0.526	0.327	0.599	0.371
	Text-Enhanced GAT	158	285	0.667	0.254	0.723	0.342	0.745	0.440
Tail entity $\langle h, r, ?\rangle$	TransH combined text model	40	44	0.582	0.401	0.597	0.536	0.648	0.575
	Text-Enhanced GAT	33	36	0.714	0.490	0.756	0.600	0.795	0.703
Average result	TransH combined text model	119	191	0.559	0.308	0.562	0.432	0.624	0.473
	Text-Enhanced GAT	96	161	0.691	0.372	0.740	0.471	0.770	0.577

[1]TransH combined text model is the baseline model.

Table III), and 10502 triples as our dataset. We randomly pick out 85% triples $(8,923)$ as the training set, 5% triples (519) as the verification set, and the rest 10% triples $(1,060)$ as the test set.

2) Hyperparameters: We implement the proposed text-enhanced graph attention network model with PyTorch. In detail, the weight attenuation coefficient of the GAT model is set to $5e^{-6}$, and the weight attenuation coefficient of the convolutional layer is set to $1e^{-5}$. We update the parameters of the network with a learning rate $1e^{-3}$. For each convolutional layer, the dropout rate is defined as 0.3 to relieve the issue of over-fitting [25]. We perform batch training on all training data each time, i.e., the batch size is 8923 during training in order to discover the optimal value. We choose Adam algorithm [26] as optimization function with its recommended hyperparameter. All experiments are run on an NVIDIA Tesla M40 GPU server.

3) Evaluation Metrics: For each triple, we aim at predicting the missing element based on the other two elements and provide a list of candidates in order for such a missing element. We adopt Hits@1, Hits@3, Hits@5, Hits@10 to evaluate the performance of the prediction [8], [21], [27]. We also use Mean Rank (MR) and Mean Reciprocal Rank (MRR) as the evaluation metrics. The MR evaluates the average position (in the candidate order) of the correct prediction for the test dataset [8], [27]. And the MRR represents the average value of the reciprocal rank of the correct entity for each prediction task [27].

*B. **RQ1:** How effective is our text-enhanced GAT approach in security entity relationship predictions, compared with the baseline method?*

Setup. In this RQ, we investigate the prediction performance of our proposed approach based on the constructed security knowledge graph. In the given test set with 10% $(1,060)$ triples, we simulate the missing relational facts by randomly removing the head or tail entity of each test triple to generate a predict-head (or tail) task — $\langle ?, r, t\rangle$ or $\langle h, r, ?\rangle$. The real head entity h or tail entity t in the tripe $\langle h, r, t\rangle$ is regarded as the correct one to be predicted. We evaluate all experimental settings in both *Filter* and *Raw* [9], [27]. For the type of *Filter*, we delete the corrupt triples that have been in the test set, while such corrupt triples are retained for later prediction in the type of *Raw*. In general, *Filter* has a lower MR value and a larger Hits@10 value [27]. Compared with the baseline (i.e., the TransH combined text model) in [9], we calculate the evaluation metrics in Section IV-A for link prediction tasks, which includes both predict-head and predict-tail tasks with overall performance.

Results. *Overall Performance.* Table IV presents the overall performance of the link prediction task compared with the baseline TransH combined text model [9], by using the same dataset listed in Section IV-A. We concentrate on the experimental results with the type of *Filter*. Because they have the same conclusions on the experiments to evaluate the improvement of the proposed model compared with baseline. As is shown in Table IV, the difference between *Filter* and *Raw* are just the former has lower MR value, larger MRR value and larger Hits@N value (N is 5 or 10). So all of the followed experiments are conducted with the *Filter* type.

We first analyze the average results on four metrics MR, MRR, Hits@5, and Hits@10 in Table IV. Our text-enhanced GAT model finds the correct entity to be predicted more early than the baseline, which is decreased from 119 to 96 under the setting of *Filter*. It also rises 0.132 MRR value from 0.559 to 0.691. From the aspect of Hits@N, we observe that the result of Hits@5 and Hits@10 increase 0.178 (from 0.562 to 0.740), 0.146 (from 0.624 to 0.770), respectively, when taking the *Filter* setting. This infers that the text-enhanced GAT model can not only represent the textual description of an entity but also extract more advantageous relations from the $2-$hop neighbors. Similarly, the predict-head and predict-

tail tasks also benefit from the proposed model. The value of MRR, Hits@5, and Hits@10 are improved by 0.131, 0.197, and 0.146 when predicting the missing head entity. And for the task of predict-tail task, MRR, Hits@5, and Hits@10 grow by 0.132, 0.159, and 0.147 respectively. Therefore, compared with baseline, our model promotes the prediction results of predict-head and predict-tail tasks with similar improvement.

The text-enhanced GAT model outperforms the state-of-the-art [9] on the overall performance for both the predict-head task and the predict-tail task.

C. *RQ2: Can our approach predict well in each type of security entity relationship?*

Setup. We further explore the model performance of link prediction tasks for each defined relation type listed in Table III with the metrics Hits@N values.

Results. *Performance of Link Prediction for Each Relation Type.* As is shown in Table V, the performance of our method varies across different relation types and across different prediction tasks. Table III presents 5 types of relations which have more than $1,000$ relations (i.e., *ChildOf*, *ParentOf*, *AttackOf*, *TargetOf*, *BelongOf*). 4 types out of them achieve high Hits@N values for both predict-head task (0.766-0.953) and predict-tail task (0.627-0.953) except *BelongOf* with the most relations $4,003$. For the *BelongOf*, it gains high Hits@N values larger than 0.621 for the predict-tail task, while it can hardly predict the correct head entity with Hits@N values even lower than 0.214. The rest four relations types *CanPrecede*, *CanFollow*, *PeerOf* and *Semantic* have few relations lower than $1,000$. The *CanPrecede* and *CanFollow* also get high (larger than 0.625) Hits@N for both prediction tasks. However, the results of *PeerOf* and *Semantic* are different with different N in the Hits@N. For example, although the prediction performance of Hits@10 values is more than 0.645, the Hits@1 values are below-0.5 (even only 0.125 for the *PeerOf* on the predict-head task).

Three factors are discussed below that make effects on the prediction results of Table V. Firstly, more triples trained for one relation type will provide more entity relation knowledge which then improves the results. Thus, the relation types with sufficient data (i.e., *ChildOf*, *ParentOf*, *AttackOf*, *TargetOf*) gives better Hits@N values, while the *PeerOf* relation is relatively low. However, due to the diversity of mapping between the CVE (head) and CWE (tail) entity ("many-to-one" for "CVE-to-CWE)" in the relation type *BelongOf*, it is difficult to find the correct CVE (head) entity based on the knowledge from only one CWE (tail) entity. So the Hits@N values for the combination of *BelongOf* and predict-head task are very low. On the opposite, the predict-tail task gives high Hits@N values because more CVE (head) entities produce ample structural and semantic information to search for the correct CWE (tail) entity. For the types *CanPrecede* and *CanFollow* with few triples 206 and 162 respectively, they also has high Hits@N values. The reason may be the stability of both textual description and relevant structure of such types. The *CanPrecede* always denotes the triples with preceding relation. For instance, *CWE-1260* precedes *CWE-119* under the description "The product allows address regions to overlap, which can result in the bypassing of intended memory protection". The *CanFollow* means the following relation, such as *CAPEC-491* follows *CAPEC-228* with the description "An adversary exploits a few properties of XML(substitution entities and inline DTDs) to cause a denial of service situation".

For the constructed 9 types of relations, it is better to collect more relations to generate a well-predicted model. Meanwhile, the diversity of triples also affects the result of link prediction, such as the relation type BelongOf with very low Hit@N values although it has the most training relations.

D. *RQ3: How sensitive can our approach learn from the complexity of structural and textual embedding compared with the ablation studies?*

Setup. We conduct an ablation study to further investigate the impact of our approach after omitting the part of CNN layers, TransE initialization, and attention layers. Then 3 revised models (i.e., M-1, M-2, M-3 for the mentioned three revisions respectively) are created to test the link prediction performance with metrics MR, MRR, Hits@1, Hits@3, Hits@5, and Hits@10.

Results. Table VI shows the results of 3 revised models (i.e., M-1, M-2, and M-3) compared with our source model. We identify that removing CNN layers, the M-1 model, suffers the least negative influence. It rises 56 MR to 152, decreases 0.068 MRR to 0.623, and the Hits@N values go down $0.06 \sim 0.073$. Then, the model with the second negative influence is M-2 to remove TransE. M-2 turns MR up to 239 and the MRR value down to 0.526. The Hits@N values of M-2 are affected by the drop of $0.073 \sim 0.207$ on the basis of the source model. The attention layers are obviously the most important part of the text-enhanced GAT model. When omitting the attention layers, the M-3 model only achieves MRR 0.482 with the most decreasing of 0.209 for the source model. It also gives the largest MR 264 and the least Hits@N values in $0.401 \sim 0.583$.

In the proposed text-enhanced GAT model, the part of the attention layer greatly affects the prediction performance, by embedding the information of $2-$hop neighbor nodes into one entity to extract more textual description knowledge from the graph. The initialization of TransE also provides critical structural knowledge of this security graph.

E. *Discussion*

In this section, we present a case study of the proposed security knowledge graph, which predicts the related CWE

TABLE V
PERFORMANCE OF LINK PREDICTION FOR EACH RELATION TYPE

	Hits@N	BelongOf	AttackOf	TargetOf	ChildOf	ParentOf	CanPrecede	CanFollow	PeerOf	Semantic
Head entity $\langle ?, r, t\rangle$	@1	0.164	0.862	0.913	0.766	0.861	0.750	0.750	0.125	0.338
	@3	0.186	0.895	0.917	0.810	0.873	0.792	0.781	0.167	0.606
	@5	0.203	0.925	0.936	0.867	0.905	0.843	0.813	0.333	0.634
	@10	0.214	0.944	0.953	0.892	0.911	0.850	0.875	0.667	0.732
Tail entity $\langle h, r, ?\rangle$	@1	0.621	0.874	0.913	0.791	0.627	0.650	0.625	0.189	0.452
	@3	0.630	0.889	0.935	0.867	0.791	0.800	0.688	0.214	0.563
	@5	0.701	0.925	0.944	0.892	0.886	0.884	0.750	0.346	0.592
	@10	0.764	0.944	0.953	0.899	0.911	0.900	0.813	0.645	0.746

TABLE VI
PERFORMANCE OF DIFFERENT REVISED MODELS (THE **BOLD** IS THE BEST)

Model	Explanation	MR	MRR	Hits@1	Hits@3	Hits@5	Hits@10
M-1	Remove CNN layers	152	0.623	0.570	0.650	0.670	0.710
M-2	Remove TransE	239	0.526	0.431	0.588	0.644	0.697
M-3	Remove attention layers	264	0.482	0.401	0.496	0.527	0.583
Our Approach	Text-Enhanced GAT	**96**	**0.691**	**0.638**	**0.723**	**0.740**	**0.770**

based on a given CVE.

As is shown in Section IV-A, we obtain $4,003$ CVEs and 98 CWEs that refers to the Linux system, by July 20, 2020. Those CVEs and CWEs build $4,003$ triples with relation *BelongOf*. The head entity is CVE and the tail entity is CWE. In general, when a new software vulnerability (CVE) is reported, the security experts can not determine what weakness (CWE) leads to such vulnerability before much complex and time-consuming analysis on this vulnerability. However, if the weakness of this vulnerability is identified in time, experts can work out a complete solution to repair it more quickly [28]. For example, in the triple <*CVE-2019-14970*, *BelongOf*, *CWE-119*>, the *CVE-2019-14970*[2] allows attackers to trigger a heap-based buffer overflow by a crafted ".mkv" file, and the relevant *CWE-119*[3] points out the buffer overflow weakness comes from the operation to read from or write to a memory location outside of the intended boundary of the buffer. After connecting the *CVE-2019-14970* and the *CWE-119* with a relation *BelongOf*, we can acquire more detailed information about what the ".mkv" file executes to result in the buffer overflow. Then security experts take advantage of such security knowledge to take a series of measures to promptly repair the software vulnerability.

Based on the constructed security knowledge graph, our experiments in Section IV-B stimulate the missing weakness (CWE) of the given vulnerability (CVE) on the 15% triples in the test set. The text-enhanced GAT model trained by 80% triples in the training set achieves high Hits@N values in $0.621 \sim 0.764$ for the task to predict the tail entity CWE (see in Table V). This implies the security knowledge graph yields a large number of constructively structure and representation knowledge to label the vulnerability (CVE) with a relevant weakness (CWE). It narrows the time window to find the unknown weakness of a newly reported software vulnerability, which effectively accelerates the analysis of this vulnerability and profit for the construction of the relevant patch.

[2]*CVE-2019-14970*, https://www.cvedetails.com/cve/CVE-2019-14970/.
[3]*CWE-119*, https://cwe.mitre.org/data/definitions/119.html.

V. RELATED WORK

A. *Security Database-based Research*

Due to the security databases (see in Section II) containing a large number of security knowledge of software vulnerabilities, weaknesses and attack patterns, many studies in recent years have concentrated on how to dig out such security knowledge for assisting security analysis and software protection. [29]–[33] focused on exploring the CVE database and predicting software vulnerability severity. Based on the description of the vulnerability report and many other functions (e.g., CVSS score, confidentiality impact), Bozorgi et al. [29] trained an ML classifier to predict whether the vulnerability can be exploited and when it will be exploited. Li et al. [30] designed a mining approach to analyze and obtain the critical software vulnerability characteristics according to three repositories (i.e., CVE, CWE, and NVD databases). To predict the vulnerability severity, [31] and [32] used deep learning to train a multi-classifier only depending on the CVE textual description. Guo et al. [13], [33] observed that CVE description can be divided into 6 aspects and many CVE descriptions are incomplete. They adopted the Bi-directional LSTM network and attention mechanism to predict the missing aspects for each CVE.

Besides, many works [8], [9] have been concerned with the CWE and CAPEC databases. By using the knowledge graph embedding method, Han et al. [8] applied TransE to perform reasoning tasks via embedding common software weaknesses and their relationships into a neural vector. Xiao et al. [9] constructed a software security heterogeneous knowledge graph with the CVE, CWE, and CAPEC databases. They adopted TransH [34] to transform the knowledge graph as a continuous vector space and complete the link prediction task between heterogeneous knowledge graphs.

Different from the literature above, we not only explore all those security databases, but also concentrate on the additional knowledge coming from the $2-$hop neighbor nodes in the graph, instead of only $1-$hop in the built triple. Therefore, we relieve the negative effect of the independence of each triple and gain more CVE, CWE, and CAPEC security knowledge.

B. Knowledge Graph Representation Learning

Given such security databases, it is necessary to construct a more effective and complete field-specific knowledge graph on the basis of the literature [8], [9]. This security knowledge graph not only provides plenty of software vulnerability knowledge but also prompt the analysis of software weakness, because many well-established field-specific knowledge graphs (i.e., Freebase [35], DBpedia [36], YAGO [37], and YAGO 4 [38]) have been successfully applied to many practical applications. During the implementation of those knowledge graphs, how to choose an efficient knowledge graph representation learning is a critical step, followed by a series of researches such as semantic analysis, named entity disambiguation, information extraction, and question answering. There are four categories of knowledge graph representation learning: translation distance model [15], [34], [39], semantic matching model [40]–[43], convolutional neural network model [24], [44], and graph neural network model [14], [21], [45]–[47].

The translation distance model uses a distance-based scoring function to measure the rationality by the distance of two entities. As a representative model, TransE [15] significantly reduced computational complexity with fewer parameters, while it cannot handle complex relationships. TransH [34] remedied this flaw and designed a relationship-specific hyperplane so that each entity node conducted specific characteristics in specific relationships. Taken a step further, TransR [39] continuously improved the hyperplane and gave different relationships with a relevant semantic space, which contained the mapped entities and generated the translation relationship from the head entity to the tail entity.

Compared with the translation distance model, the semantic matching model applies the similarity-based scoring function through matching the semantics knowledge of entities and the relationships in the vector space. For example, RESCAL [40] captured potential semantic interactions by using tensor products, but it cost too much effort to build and optimize a relationship model with huge parameters. The DISTMULT [41], an improved version of RESCAL, adopted the weighted element-wise dot products to represent entity relations (only the symmetric relations). Nickel et al. [42] proposed the HolE that created a more efficient composite representation approach via computing the circular correlation of entity embeddings. And Trouillon et al. [43] designed the ComplEx with complex-valued embeddings to decreases model parameters when calculating asymmetric relationships.

In recent years, convolutional neural network and graph neural network [45] have blossomed into knowledge graph representation learning such as ConvE [44] and ConvKB [24]. ConvE model used 2D convolution to reshape the head entity and relation into a $2-$dimensional matrix, without considering the tail entity, and captures the characteristics of the local relationship. ConvKB directly concatenated the embedding representation of the head and tail entities and relationships, and then used CNN with stronger feature learning capabilities to capture global features while maintaining a certain translational characteristic, thus achieving better experimental results. [46], [47] adapted the graph convolutional neural network in the field of computer vision and applied it into representation learning by stacking the convolutional layer. Velickovic et al. [14] built the graph attention neural network (GAT) based on the self-attention mechanism. GAT sets different weights to different neighboring nodes and only focuses on the most relevant parts.

Considering the effective information representation and better prediction performance, we use the text-enhanced graph attention method based on the graph attention network. Furthermore, our proposed approach extracts the $2-$hop neighbor nodes from the graph and implement relevant feature embedding methods (see in Section III) which can better predict the relations among security entities by 77.0%.

VI. Conclusion

Based on the constructed security knowledge graph of the CVE, CWE, and CAPEC datasets, we further propose a text-enhanced GAT model to predict security entity relations. This model adopts 2-hop neighbor nodes as the additional knowledge of one entity and the graph attention network to represent the structural information and the textual information of the graph. It outperforms the state-of-the-art by 0.132 (up to 0.691) with the metric MRR for the overall performance of link prediction. We also identify the advantages of more triples and the disadvantage of more diversity in the predict-head and predict-tail tasks for each type of defined relation. Furthermore, we clarify the importance of the attention layer in our model which is of great benefit to obtain the critical features for prediction. In the future, we will further explore the fine-grained semantic relations of the CWE database and expand more security databases such as the OVAL database and the Exploit database, in order to build a more complete security knowledge graph for assisting the security analysis.

Acknowledgments

This work has partially been sponsored by the National Science Foundation of China (No. 61872262).

REFERENCES

[1] S. Chen, T. Su, L. Fan, G. Meng, M. Xue, Y. Liu, and L. Xu, "Are mobile banking apps secure? what can be improved?" in *ESEC/FSE*. ACM, 2018, pp. 797–802.

[2] S. Chen, L. Fan, G. Meng, T. Su, M. Xue, Y. Xue, Y. Liu, and L. Xu, "An empirical assessment of security risks of global Android banking apps," in *ICSE*. IEEE Press, 2020, pp. 596–607.

[3] Z. Tang, K. Tang, M. Xue, Y. Tian, S. Chen, M. Ikram, T. Wang, and H. Zhu, "iOS, your OS, everybody's OS: Vetting and analyzing network services of iOS applications," in *USENIX Security*, 2020, pp. 2415–2432.

[4] X. Zhan, L. Fan, S. Chen, F. Wu, T. Liu, X. Luo, and Y. Liu, "Atvhunter: Reliable version detection of third-party libraries for vulnerability identification in android applications," in *ICSE*. IEEE Press, 2021.

[5] C. MITRE, "Common vulnerabilities and exposures [ol]," https://cve.mitre.org/, [Online; accessed 20-July-2020].

[6] CWE, "Common weakness enumeration [ol]," http://cwe.mitre.org/, [Online; accessed 20-July-2020].

[7] CAPEC, "Common attack pattern enumeration and classification [ol]," http://capec.mitre.org/, [Online; accessed 20-July-2020].

[8] Z. Han, X. Li, H. Liu, Z. Xing, and Z. Feng, "Deepweak: Reasoning common software weaknesses via knowledge graph embedding," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2018, pp. 456–466.

[9] H. Xiao, Z. Xing, X. Li, and H. Guo, "Embedding and predicting software security entity relationships: A knowledge graph based approach," in *International Conference on Neural Information Processing*. Springer, 2019, pp. 50–63.

[10] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. Association for Computing Machinery, 2014, p. 101–110.

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

[12] Y. Zhang, H. Dai, Z. Kozareva, A. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[13] H. Guo, Z. Xing, S. Chen, X. Li, Y. Bai, and H. Zhang, "Key aspects augmentation of vulnerability description based on multiple security databases," in *In COMPSAC*, 2021.

[14] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[15] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *In First International Conference on Learning Representations, Workshop Track Proceedings, Efficient Estimation of Word: IEEE*, 2013.

[17] CPE, "Common platform enumeration [ol]," http://cpe.mitre.org/, [Online; accessed 20-July-2020].

[18] C. MITRE, "National vulnerability database home [ol]," https://nvd.nist.gov/, [Online; accessed 20-July-2020].

[19] C. Details, "Cve details [ol]," http://www.cvedetails.com/, [Online; accessed 20-July-2020].

[20] Exploit-db, "Exploit database [ol]," https://www.exploit-db.com/, [Online; accessed 20-July-2020].

[21] D. Nathani, J. Chauhan, C. Sharma, and M. Kaul, "Learning attention-based embeddings for relation prediction in knowledge graphs," in *Meeting of the Association for Computational Linguistics*, 2019.

[22] Z. Lin, M. Feng, C. N. D. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *arXiv preprint arXiv:1703.03130*, 2017.

[23] X. Li, H. Jiang, Y. Kamei, and X. Chen, "Bridging semantic gaps between natural languages and apis with word embedding," in *IEEE Transactions on Software Engineering*, 2018, pp. 1–1.

[24] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," in *In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2018, p. 327–333.

[25] G. I. Webb, *Overfitting*. Springer US, 2011.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv preprint arXiv:1412.6980*, 2014.

[27] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[28] M. Aota, H. Kanehara, M. Kubo, N. Murata, B. Sun, and T. Takahashi, "Automation of vulnerability classification from its description using machine learning," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–7.

[29] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: Learning to classify vulnerabilities and predict exploits," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, p. 105–114.

[30] X. Li, J. Chen, Z. Lin, L. Zhang, Z. Wang, M. Zhou, and W. Xie, "A mining approach to obtain the software vulnerability characteristics," in *2017 Fifth International Conference on Advanced Cloud and Big Data (CBD)*, 2017, pp. 296–301.

[31] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to predict severity of software vulnerability using only vulnerability description," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2017, pp. 125–136.

[32] X. Gong, Z. Xing, X. Li, Z. Feng, and Z. Han, "Joint prediction of multiple vulnerability characteristics through multi-task learning," in *2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2019, pp. 31–40.

[33] H. Guo, Z. Xing, and X. Li, "Predicting missing information of vulnerability reports," in *Companion Proceedings of the Web Conference 2020*, 2020, p. 81–82.

[34] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *AAAI*, 2014, pp. 1112–1119.

[35] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250.

[36] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," in *Semantic web*, 2015, pp. 167–195.

[37] M. Fabian, K. Gjergji, W. Gerhard *et al.*, "Yago: A core of semantic knowledge unifying wordnet and wikipedia," in *16th International World Wide Web Conference, WWW*, 2007, pp. 697–706.

[38] T. P. Tanon, G. Weikum, and F. Suchanek, "Yago 4: A reason-able knowledge base," in *European Semantic Web Conference*. Springer, 2020, pp. 583–596.

[39] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *In Proceedings of AAAI*, 2015.

[40] M. Nickel, V. Tresp, and H. P. Kriegel, "A three-way model for collective learning on multi-relational data," in *International Conference on International Conference on Machine Learning*, 2011.

[41] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *ICLR*, 2015.

[42] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, p. 1955–1961.

[43] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International Conference on Machine Learning (ICML)*, 2016.

[44] D. Tim, M. Pasquale, S. Pontus, and R. Sebastian, "Convolutional 2d knowledge graph embeddings," in *In Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

[45] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," in *IEEE Transactions on Neural Networks*. IEEE, 2008, pp. 61–80.

[46] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2016.

[47] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.