DeepWeak: Reasoning Common Software Weaknesses via Knowledge Graph Embedding

Zhuobing Han*, Xiaohong Li*, Hongtao Liu*, Zhenchang Xing†, and Zhiyong Feng‡
*Tianjin Key Laboratory of Advanced Networking (TANK), School of Computer Science and Technology,
Tianjin University, Tianjin, China, {zhuobinghan, xiaohongli, htliu}@tju.edu.cn
†Research School of Computer Science, Australian National University, Australia, zhenchang.xing@anu.edu.au
‡School of Computer Software, Tianjin University, Tianjin, China, zyfeng@tju.edu.cn

***Abstract*—Common software weaknesses, such as improper input validation, integer overflow, can harm system security directly or indirectly, causing adverse effects such as denial-of-service, execution of unauthorized code. Common Weakness Enumeration (CWE) maintains a standard list and classification of common software weakness. Although CWE contains rich information about software weaknesses, including textual descriptions, common sequences and relations between software weaknesses, the current data representation, i.e., hyperlined documents, does not support advanced reasoning tasks on software weaknesses, such as prediction of missing relations and common consequences of CWEs.Such reasoning tasks become critical to managing and analyzing large numbers of common software weaknesses and their relations. In this paper, we propose to represent common software weaknesses and their relations as a knowledge graph, and develop a translation-based, description-embodied knowledge representation learning method to embed both software weaknesses and their relations in the knowledge graph into a semantic vector space. The vector representations (i.e., embeddings) of software weaknesses and their relations can be exploited for knowledge acquisition and inference.We conduct extensive experiments to evaluate the performance of software weakness and relation embeddings in three reasoning tasks, including CWE link prediction, CWE triple classification, and common consequence prediction. Our knowledge graph embedding approach outperforms other description- and/or structure-based representation learning methods.**

***Index Terms*—Software weakness prediction, multi-class classification, knowledge graph, mining software repositories.**

I. Introduction

Common software weaknesses refers to issues in software requirements, design, implementation, and/or operation that may have direct or indirectly impact on software security. For example, *integer overflow or wraparound*[1] is a common software weakness. It refers to the issue that "the software performs a calculation that can produce an integer overflow or wraparound, when the logic assumes that the resulting value will always be larger than the original value." *Integer overflow or wraparound* may be exploited to harm the system. Common consequences may include *denial-of-service* or *execute unauthorized code or commands*. A software weakness is often related to other software weaknesses. For example, *integer overflow or wraparound* is a specific case of the more general software weakness *improper input validation*[2]. It may be caused by *incorrect conversion between numeric types*[3] because developers misunderstand the programming language's numeric calculation, and it may further cause *improper restriction of operations within the bounds of a memory buffer (see Fig. 1)*.

Organizations want assurance that the software products they acquire and develop are reviewed for known types of security flaws. This demands a taxonomy or standard to define the capabilities and coverage of the tools and services that can be used for this type of review. Common Weakness Enumeration (CWE) [1] was created specifically to address this need. CWE is a community-developed standard list and classification of common software weaknesses, which serves as a unifying language of discourse and as a baseline standard for weakness identification, mitigation, and prevention efforts. In fact, the above-mentioned examples of software weaknesses are from the CWE list (see the relevant footnotes).

CWE data contains rich information about software weaknesses. In this work, we focus on three pieces of information: 1) the title and description of a CWE, 2) the common consequences of a CWE, and 3) the relations between CWEs. Fig. 1 shows an example of these three pieces of information of the *CWE-190 Integer Overflow or Wraparound*. CWE data is stored in the form of hyper-linked documents that can be accessed using information retrieval techniques. However, hyper-linked documents do not support advanced reasoning tasks on the CWE data. For example, *CWE-203: Information Exposure Through Discrepancy* may cause "read application data" and "bypass protection mechanism". But these two consequences were missing in the version 1.0 of CWE data and was added in the version 2.0. As another example, *CWE-190: Integer Overflow or Wraparound* can precede *CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer*. But the link between the two CWEs was missing until the version 2.0. Unfortunately, hyper-linked documents do not support the inference of common consequences or the prediction/derivation of CWE relations.

To support reasoning over common software weaknesses, we propose to represent CWE data as a knowledge graph of common software weaknesses. Each CWE is an entity in the knowledge graph, and it has two attributes - *textual*

[1]https://cwe.mitre.org/data/definitions/190.html
[2]https://cwe.mitre.org/data/definitions/20.html
[3]https://cwe.mitre.org/data/definitions/681.html

descriptions and *categorical consequences*. Textual descriptions can be of different level of details. Relations between CWEs include CWE predefined relationships including *parent-child*, *precede-follow*, *peerof*, and general semantic relatedness extracted from discussions of CWE demonstrative examples and potential mitigations (see Fig. 1).

Reasoning over knowledge graphs traditionally adopts symbolic representation of knowledge graphs and graph-based methods (e.g., Path Ranking Algorithm [2], Question Answering Task [3], Temporal Information Reasoning [4]) for link prediction and Horn rule extraction tasks. These methods cannot effectively incorporate structure knowledge of CWEs with textual knowledge of CWEs. And they cannot deal with zero-shot scenario, where some CWEs are novel to existing knowledge graphs with only descriptions. Recently, representation learning for knowledge graphs has been proposed to embed knowledge graphs including both entities and relations into a continuous low-dimensional vector space [5], [6], [7], and the entity and relation embeddings have demonstrated the advantages over traditional symbolic representation for knowledge acquisition and inference.

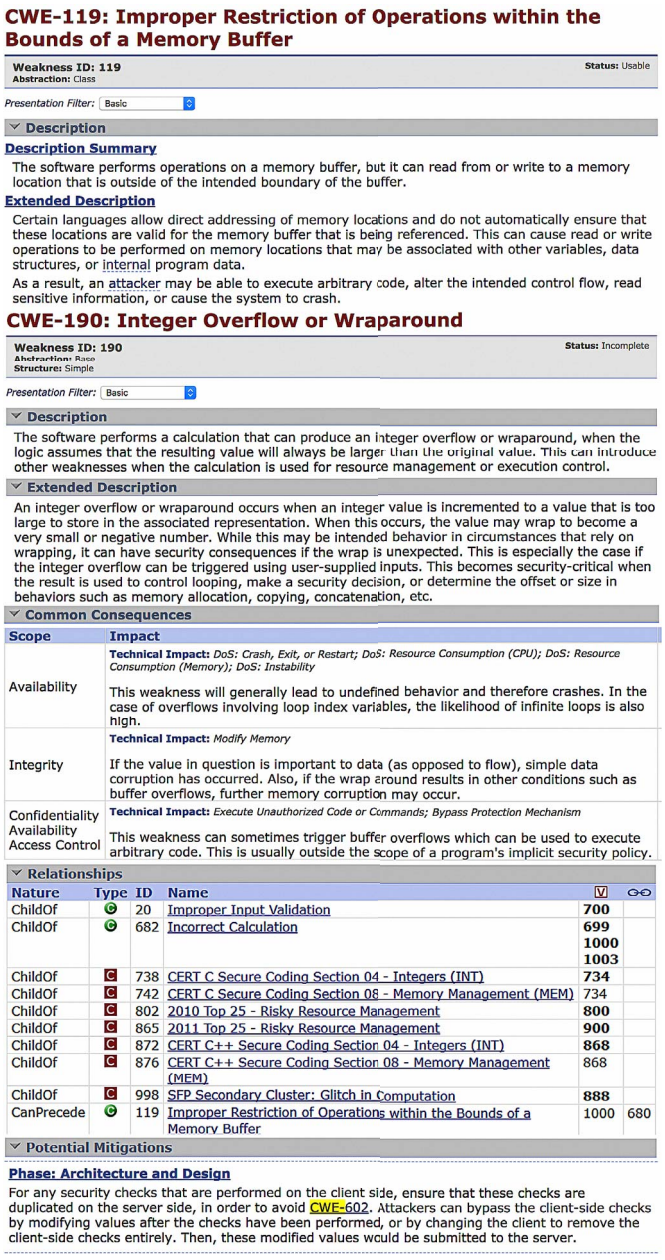
In this paper, we adopt translation-based knowledge graph embedding approach for embedding CWE knowledge graph. Translation-based methods consider a knowledge graph as triple facts (*head entity, relation, tail entity*), and interpret relations as translating operations between head and tail entities. Meanwhile, we explore neural network encoder to represent semantics of CWE descriptions. The entity and relation embeddings are learned to maximize the likelihood of relation translations as well as the likelihood of predicting a CWE's description.

We download the CWE data from its official website. We build a parser to convert hyper-linked CWE documents into a knowledge graph of CWEs. We evaluate the effectiveness of our CWE knowledge graph embedding model on three tasks, including CWE relation and entity ranking, CWE triplet classification, and CWE common consequences prediction. Our experiment results confirm the effectiveness of our knowledge graph embeeding approach, compared with description-only, structure-only and description-structure-vector-addition based representation learning methods.

This work makes the following contributions:

- We are the first to investigate the knowledge graph representation of common software weaknesses and the reasoning tasks on this knowledge graph.
- We propose a knowledge graph embedding approach to embed the structural and textual knowledge of CWEs into vector representations.
- We conduct extensive experiments to evaluate the effectiveness of knowledge graph embeddings in reasoning over relations and common consequences of CWEs.

The remainder of this paper is organized as follows. Section II presents the background of the CWE and knowledge graph embedding. Section III elaborates the technical details of our approach. Section IV reports our experiments and findings.



CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer

Weakness ID: 119 **Status:** Usable
Abstraction: Class

Presentation Filter: Basic

Description

Description Summary

The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.

Extended Description

Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data.

As a result, an attacker may be able to execute arbitrary code, alter the intended control flow, read sensitive information, or cause the system to crash.

CWE-190: Integer Overflow or Wraparound

Weakness ID: 190 **Status:** Incomplete
Abstraction: Base
Structure: Simple

Presentation Filter: Basic

Description

The software performs a calculation that can produce an integer overflow or wraparound, when the logic assumes that the resulting value will always be larger than the original value. This can introduce other weaknesses when the calculation is used for resource management or execution control.

Extended Description

An integer overflow or wraparound occurs when an integer value is incremented to a value that is too large to store in the associated representation. When this occurs, the value may wrap to become a very small or negative number. While this may be intended behavior in circumstances that rely on wrapping, it can have security consequences if the wrap is unexpected. This is especially the case if the integer overflow can be triggered using user-supplied inputs. This becomes security-critical when the result is used to control looping, make a security decision, or determine the offset or size in behaviors such as memory allocation, copying, concatenation, etc.

Common Consequences

Scope	Impact
Availability	**Technical Impact:** *DoS: Crash, Exit, or Restart; DoS: Resource Consumption (CPU); DoS: Resource Consumption (Memory); DoS: Instability* This weakness will generally lead to undefined behavior and therefore crashes. In the case of overflows involving loop index variables, the likelihood of infinite loops is also high.
Integrity	**Technical Impact:** *Modify Memory* If the value in question is important to data (as opposed to flow), simple data corruption has occurred. Also, if the wrap around results in other conditions such as buffer overflows, further memory corruption may occur.
Confidentiality Availability Access Control	**Technical Impact:** *Execute Unauthorized Code or Commands; Bypass Protection Mechanism* This weakness can sometimes trigger buffer overflows which can be used to execute arbitrary code. This is usually outside the scope of a program's implicit security policy.

Relationships

Nature	Type	ID	Name	V	
ChildOf	B	20	Improper Input Validation	**700**	
ChildOf	B	682	Incorrect Calculation	**699** **1000** **1003**	
ChildOf	C	738	CERT C Secure Coding Section 04 - Integers (INT)	**734**	
ChildOf	C	742	CERT C Secure Coding Section 08 - Memory Management (MEM)	734	
ChildOf	C	802	2010 Top 25 - Risky Resource Management	**800**	
ChildOf	C	865	2011 Top 25 - Risky Resource Management	**900**	
ChildOf	C	872	CERT C++ Secure Coding Section 04 - Integers (INT)	**868**	
ChildOf	C	876	CERT C++ Secure Coding Section 08 - Memory Management (MEM)	868	
ChildOf	C	998	SFP Secondary Cluster: Glitch in Computation	**888**	
CanPrecede	B	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	1000	680

Potential Mitigations

Phase: Architecture and Design

For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.

Fig. 1. An excerpt of the CWE-190 information

Section V reviews related work. Section VI summarizes our contributions and our plans for future work.

II. Background

A. Common Weakness Enumeration (CWE)

CWE [1] provides a list of common software security weaknesses, each of which is a class of bug or flaw in softwares. It serves as a baseline for weakness identification, mitigation, and prevention efforts and also could be used for measuring software security tools. There are totally 1005 CWEs in the latest version of CWE (version-2.11) including four types:

TABLE I
STATISTICS OF CWE RELATIONSHIPS IN CWE (VERSION-2.11)

Relation Type	parent child	follow precede	membe of	peer of	requires	can_also be	start with
Number	3384	121	365	142	17	34	3

- *Weakness* is a type of mistake in software which could be exploited by attackers resulting in vulnerabilities.
- *Category* is a CWE contains a set of other CWEs that share a common characteristic.
- *View* is a subset of CWEs that provides a way of examining CWE content in the structure of slices and graphs.
- *Compound Element* is a CWE that closely associates two or more CWE entries.

Weakness is atomic, while category, view and compound element are different ways of grouping weaknesses. In this work, We deal with only the 705 weaknesses such as buffer overflow, directory traversal, OS injection, cross-site scripting, and insecure random numbers. We leave the study of different grouping mechanisms as our future work.

In each weakness's content, several pieces of information are provided such as CWE id, CWE title, description, common consequences, detection methods, demonstrative examples, potential mitigations, and relationships (as shown in Fig. 1). In this work, we extract CWE id, title, description summary, common consequences, and relationships from each weakness's content. Each CWE is regarded as an entity which is identified by its id, and has a textual description (title+description summary) and several categorical common consequences.

According to the number of different types of CWE relationships listed in Table I, we decide to use the most common relations to build the CWE knowledge graph, i.e., *parent-child*, *precede-follow*, and *peerof*. Since we only take weakness into consideration, the relation *memberof* which is related to the category and view CWEs is not included. The *parent-child* relationship is easy to understand, which represents the generalization-specialization relationship. The *precede-follow* relationship is used to identify when the weakness is primary to or resultant from others. The *peerof* relationship stands for the equally privileged relation between CWEs. Besides, the forth type of relationship *semantically related* is extracted from hyperlinked CWEs discussed in demonstrative examples and potential mitigations sections. In total, 871 semantically-related relationships have been extracted.

B. Knowledge Graph Embedding

A variety of knowledge graphs have been created for ranking, reasoning and classification tasks, because knowledge graphs use relations between entities to describe facts in the world. Knowledge graph embeddings can projection a large scale knowledge graph into a continuous low-dimension vector space. There are many methods for modeling multi-relational data in knowledge graphs, such as TransE [5], TransH [6], TransD [8], TransR [7]. They are promising translation-based methods that achieve state-of-the-art predictive performance. TransE is a basic model of knowledge graph embedding model and also the basis of our approach. TransE interprets the relations as translating operations between head and tail entities on the low-dimensional vector space. The energy function is defined as $E(h, r, t) = \|h + r - t\|$, which indicates that the tail embedding t should be the nearest neighbour of $h + r$.

III. THE APPROACH

In this section, we formulate the research problem, present an overview of our approach and describe the technical details.

A. Problem Formulation

This work aims at designing an effective knowledge graph embedding approach to support reasoning over common software weaknesses based on the relationships between CWEs and the description of each CWE. Specifically, we formulate the task of CWE knowledge graph embedding into learning CWE embeddings and CWE relation type embeddings. The learned knowledge graph embedding model can be used to embed CWEs and CWE relations into a vector space in order to support reasoning tasks such as CWE relationship prediction and CWE common consequences prediction.

B. Translation-based Knowledge Graph Embedding with Entity Descriptions

The input to our approach is a knowledge graph of CWEs. Each CWE relation in the knowledge graph is represented as a triplet $(h, r, t) \in T$, where $h, t \in V$ denoting the head CWE entity and tail CWE entity respectively and $r \in R$ denoting the relationship type. V is the set of CWE entities and R is the set of CWE relationship types. In this work, we consider 4 types of CWE relationships: *parent-child*, *precede-follow*, *peerof* and *semantically related*. T stands for the training set of triplets for learning entity and relation embeddings.

The embeddings of entities and relations take values in $\mathbb{R}^k$, i.e., k-dimensional real-value vectors. Each entity will be embedded in two representations: structure-based representation and description-based representation. h_s and t_s are the structure-based representations for the head entity and tail entity learned from CWE relations. h_d and t_d are the description-based representations for the head and tail entity learned from the corresponding CWE descriptions.

Fig. 2 presents the overall architecture of our TransE-based [5] Concatenation Model, called TransCat, for embedding the CWE knowledge graph. The general idea of our approach is to learn the structure representation of an entity as well as the description representation from a graph of known CWEs. We first crawl relations between CWEs and CWE descriptions from the CWE website [1]. We preprocess the relationships by merging the symmetrical relations like "ChildOf" and "ParentOf" as "parent-child", "CanPrecede" and "CanFollow"as "precede-follow". CWE descriptions are also preprocessed by removing digits, punctuation and some special symbols like "*", "#", "@", "//", "/./". The resulting knowledge graph contains triplets in the form of ⟨*head-entity description, relation type, tail-entity description*⟩

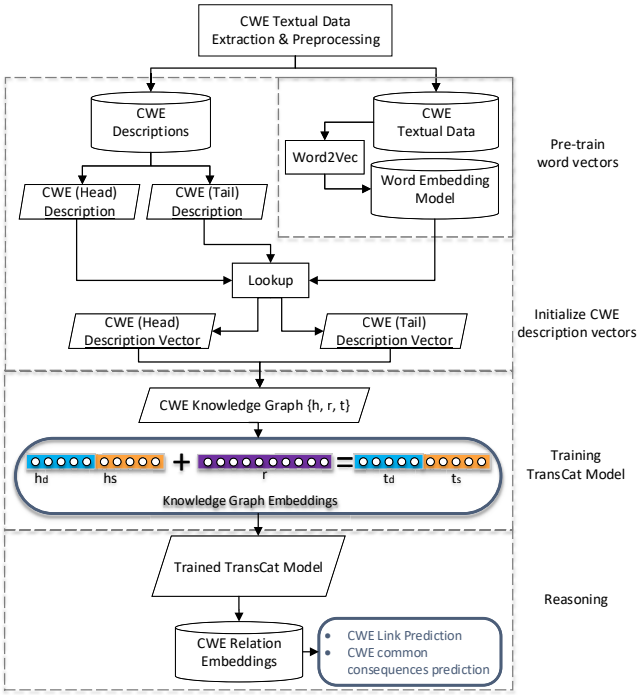


Fig. 2. Overview of our approach

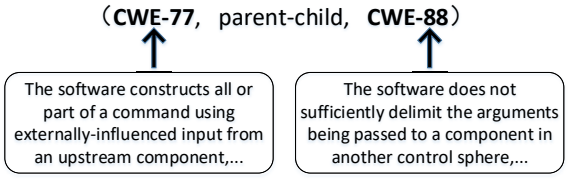


Fig. 3. A Triplet Example in the Knowledge Graph of CWEs

for each CWE relation. A triplet example is shown in Fig. 3, which represents a parent-child relation between the CWE-77 and the CWE-88.

To embed the CWE descriptions, We first learn word representation by word embeddings [9] which can capture rich syntactic and semantic features of words. The corpus used for training word embedding is all the text crawled from the CWE website including not only CWE descriptions but also the descriptions of demonstrative examples and potential mitigations. We use the continuous skip-gram model [10] to train the word embedding and the output is a dictionary of word vectors for each word in the CWE description vocabulary. After learning word level representations, we still need to capture sentence level features in order to represent the CWE description. To that end, We use the average vector of the word embeddings of all the words in a CWE description as the sentence level representation of a CWE.

When description-based representation of CWEs is learned, structure-based representations of CWEs are learned by the knowledge graph embedding method TransE [5]. Both description-based and structure-based representations will be trained jointly under the translation-based method. The energy function is defined as Eq (1).

$$E = E_S \oplus E_D \tag{1}$$

where E_S is the same energy function as TransE [5], i.e., $E_s = \|h_s + r - t_s\|$. E_d is a new-added part standing for the description-based representations which is defined as $E_d = \|h_d + r - t_d\|$. The symbol $\oplus$ represents the concatenation of two representations of entities. The objective of knowledge graph embedding is to minimize E over the training set of triplets T. Recall that translation-based knowledge graph embedding method considers relations as translating operations between head and tail entities. Intuitively, $E_s = \|h_s + r - t_s\|$ indicates that the resulting vector of adding the structure-based vector of the head CWE and the vector of the relationship type should be close (by Eucliden distance) to the structure-based vector of the tail CWE. Same interpretation can be applied to $E_d = \|h_d + r - t_d\|$

C. Learning Description-based Representation of CWEs

The first step of our approach is to represent CWE descriptions as vectors to form the description-based representations of CWEs. Each word in the CWE description is a discrete symbol which cannot be accepted by a model directly. Thus, representing words into a vector space is highly recommended. Considering the vocabulary size of CWE descriptions, one-hot vector [11] will be of very high dimension and the resulting representation of CWE descriptions will be very sparse. Therefore, we adopt word embeddings which are a dense low-dimensional vector representation of words to capture semantic and syntactic features of words for NLP tasks [9], [12], [13].

Based on an assumption that words appear in similar context tend to have similar meanings, word embeddings are usually computed using neural language models. Continuous skip-gram model [9], [10] is used in this paper. The general idea of continuous skip-gram model is to predict the surrounding words with a current word. The objective function of the model is to maximize the sum of log probabilities of the surrounding context words w_{i+j} conditioned on the current word w_i:

$$\sum_{i=1}^{n} \sum_{-k \le j \le k, j \neq 0} \log p\left(w_{i+j} | w_i\right) \tag{2}$$

where n stands for the length of the word sequence and $2k+1$ is the size of context window. The conditional probability $p\left(w_{i+j}|w_i\right)$ is defined by the softmax function as Eq (3) and computed by negative sampling method [10].

$$p\left(w_{i+j}|w_i\right) = \frac{\exp\left(u_{w_{i+j}}^T v_{w_i}\right)}{\sum_{w \in W} \exp\left(u_w^T v_{w_i}\right)} \tag{3}$$

Word embeddings need to be trained by large amount of text. To learning common software weakness related word embeddings, we crawl all the text from CWE website to form a common software weakness specific corpus. The trained word embeddings cover all words in CWE descriptions. A dictionary of word embeddings for each word in the description

vocabulary can be obtained after training the model. Each word in a CWE description is associated with a word vector by looking up the dictionary of word embeddings. A CWE description with n words is then represented as a set of n word vectors [14], [15]. We take the average of n word vectors as the sentence vector, i.e., the description-based representation of a CWE entity h_d.

D. Learning Structure-based Representation

After learning the description-based representations of CWE entities, the structure-based representation is to be learned. Since the goal of learning structure-based representation is to encode both CWE entities and the relation between CWEs into a continuous low-dimensional vector space, our approach adopts the recently proposed translation-based methods of modeling multi-relational data in knowledge graphs, especially the TransE model [5]. TransE can interpret the CWE relations as translating operations between the head CWE entity and the tail CWE entity on the low-dimensional vector space. In our approach, the entity dimension of knowledge graph embeddings are the same as word embeddings, the relation dimension is the double of them.

For a given CWE relation $(h, r, t) \in T$, where $h, t \in V$ denoting the head CWE entity and tail CWE entity respectively and $r \in R$ denoting the relation. V is the set of entities and R is the set of relationships. Then the energy function E of TransE is defined as Eq (4).

$$E(h, r, t) = \|h + r - t\| \tag{4}$$

The general idea of this model is that the functional relation induced by the r labeled edges corresponds to a translation of the embeddings, i.e., $h + r \approx t$. The geometric interpretation is that the tail embedding t should be the nearest neighbour of the resulting vector of vector addition $h + r$.

By learning the structure-based representation E_s and description-based representation E_d, we concatenate the two types of vector embeddings to build the final TransCat embedding model of the CWE entities as $E = E_S \oplus E_D$, where $E_s = \|h_s + r - t_s\|$, $E_d = \|h_d + r - t_d\|$.

E. Model Initialization and Optimization

The objective of our TranCat model is to minimize the margin-based score function, i.e., the loss function:

$$L = \sum_{(h,r,t)\in T} \sum_{(h',r,t')\in T'} \max\left(\gamma + E\left(h, r, t\right) - E\left(h', r, t'\right), 0\right) \tag{5}$$

where $\gamma > 0$ is a margin hyperparameter. $E(h, r, t)$ is the energy function defined above. T' is the negative sampling set of T, which is defined as:

$$T' = \{(h', r, t)\,|h' \in E\} \cup \{(h, r, t')\,|t' \in E\} \tag{6}$$

in which the head or tail CWEs are randomly replaced by another CWE (but not both at the same time) which does not have the relation (h', r, t) or (h, r, t'). This margin-based score function L essentially requires that the gap between the energy score of present relations and that of non-present relations should be greater than the margin γ.

Algorithm 1: Learning TransCat

Input: Training set $T = \{(h, r, t)\}$, entity set V and relation set R, margin γ, embeddings dimension k.
Output: Knowledge graph embedding model.

Let $r \leftarrow uniform\left(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}\right)$ for each relation $r \in R$;
Let $r \leftarrow \frac{r}{\|r\|}$ for each relation $r \in R$;
Let $e \leftarrow uniform\left(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}\right)$ for each entity $e \in V$;
foreach $e \in V$ **do**
    Let $e \leftarrow \frac{e}{\|e\|}$;
    Let $e \leftarrow e \uplus d$; // d is the description-based embedding
    $T_{batch} \leftarrow \emptyset$; // initialize the set of pairs of triplet
    $T_{batch} \leftarrow T_{batch} \cup \{((h, r, t), (h', r, t'))\}$;
Update embedding
$$\sum_{((h,r,t),(h',r,t'))\in T_{batch}} max\left(\gamma + E(\mathbf{h}, \mathbf{r}, \mathbf{t}) - E(\mathbf{h}', \mathbf{r}, \mathbf{t}'), 0\right)$$

Finally, the Adam update algorithm [16] is adopted to optimize the loss function of our TransCat model. A constraint is carried out that the L2-norm of the embeddings of all the entities is 1 which is the same as in TransE. The optimization procedure is detailed in Algorithm 1. It first initializes all the embeddings for entities and relations randomly as in [17] and then normalize the embedding vectors of each entity at the beginning of each iteration. Specifically, the $uniform$ standards for the uniform distribution in the interval $[-1, 1]$, and k is the dimension of the embeddings. $x/\|x\|$ is the operation of normalization, where $\|x\|$ means the norm operator. The algorithm then concatenates the description-based vector with the normalized embedding vectors to form the new embedding vectors of each CWE. Next, it initializes a mini batch of pairs of triplet from training set and make a negative sample of the triplet. The algorithm updates the parameters with constant learning rate and it stops according to the performance on a validation set.

IV. Experiment

In this section, we empirically study and evaluate our knowledge-graph embedding approach on three reasoning tasks using the learned entity and relation embeddings: CWE relationship ranking, CWE triplet classification, and CWE common consequences prediction. Several experiments are conducted to make comparison with other CWE representation learning methods. The hyperparameters of the word embedding and TransCat model are optimized and the training cost of our approach is also reported in this section. All experiments run with PyTorch on an NVIDIA GTX 960M GPU.

A. Dataset

We prepare the experimental dataset from the latest version of the CWE data (V2.11). For each CWE, we crawl CWE title and description summary, which allows us to investigate the impact of description details on the CWE embeddings. The Statistics of the CWE basic dataset are listed in Table II. We randomly select 85% triplets for training the knowledge graph

TABLE II
STATISTICS OF BASIC DATASET

#CWE	#Triplets	#RelType	#ComConseqType
705	3868	4	8

TABLE III
STATISTICS OF DATASET FOR KNOWLEDGE GRAPH EMBEDDING AND LINK PREDICTION

#CWE	#RelType	#Train	#Valid	#Test
705	4	3288	193	387

TABLE IV
STATISTICS OF DATASET FOR COMMON CONSEQUENCES PREDICTION

#CWE	#ComConseqType	#Train	#Test
523	8	471	52

TABLE V
HYPERPARAMETERS OPTIMIZATION

Batch size	Hits@1(%)	Margin γ size	Hits@1(%)	Embedding Dimension	Hits@1(%)
16	**0.853**	2.0	0.811	32	0.834
32	0.824	**5.0**	**0.853**	**64**	**0.853**
64	0.829	8.0	0.824	128	0.826

embedding model and 5% triplets for model hyperparameters optimization.

For the CWE link prediction task, we crawl four types of CWE relations, including the CWE relations provided by the CWE website (*parent-child*, *precede-follow* and *peerof*), and general semantic relatedness extracted from the discussions on CWE demonstrative examples and potential migrations. We use the rest 10% triplets for testing the performance of the CWE and CWE-relation embeddings obtained from the trained TransCat model for CWE relationship ranking task (see Table III). For the CWE triplet classificiation task, same training and testing triplets are used as in the CWE relationship ranking task for training and testing the triplet binary classifier.

For the CWE common consequences prediction task, we crawl the technical impacts of common consequences of each CWE. Overall, we collect 8 types of common consequences, including *Read data, Modify memory, Denial-of-Service: unreliable execution, Denial-of-Service: resource consumption, Execute unauthorized code or commands, Gain privileges/assume identity, Bypass protection mechanism, Hide activities*. We select 90% triplets for training the common consequences multi-lablel classifier and 10% triplets for testing (see Table IV).

B. Baseline Models

We design three baseline methods for the experiments. All the baseline methods are using either the description or the structure representations of CWE dataset as the input, including 1) word embedding features; 2) only structure embeddings of knowledge graph; and 3) the joint feature of word embedding and structure embeddings of knowledge graph. The baseline methods are trained by the same training set as that for training our TransCat. And they use the same setting of hyperparameters as in our TransCat (see Section IV-C).

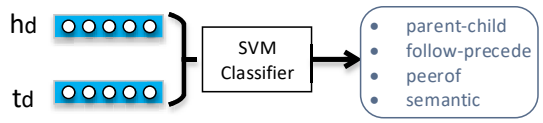


Fig. 4. Baseline 1: Word Embedding and SVM (Only Description).

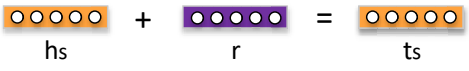


Fig. 5. Baseline 2: Knowledge Graph Embedding (Only Structure).

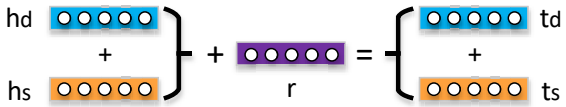


Fig. 6. Baseline 3: Knowledge Graph Embedding (Structure+Description)

1) Baseline 1: word embedding + SVM (only description): The Baseline 1 is designed to extract semantic features of words in CWE descriptions by word embeddings. The architecture of baseline 1 is showed in Fig. 4. The CWE descriptions are represented by taking the average of the word embeddings of the words comprising the description, which is the same as our approach. In this baseline, we also use the same setting of word embedding dimension as in our TransCat. Therefore, the CWE description vector is a 64-dimensional vector as the input to the SVM classifier. Since this baseline only takes description-based representation into consideration, it can be adopted only in the relation ranking task instead of other CWE reasoning tasks.

2) Baseline 2: TransE (only structure): The Baseline 2 is designed to extract structure-based features in CWE entities and relationships by the basic knowledge graph embedding method, i.e., TransE, which is commonly used as the baseline method in knowledge graph embedding tasks [18], [7]. Compared with our approach, this baseline method only take CWE structure information into consideration. The architecture of baseline 2 is showed in Fig. 5. We use the same setting of hyperparameters as in our TransCat. Our implementation of TransE performs better than results reported in [5].

3) Baseline 3: TransE (structure + description): The Baseline 3 is designed to extract both the description-based and structure-based features in CWE descriptions and relationships by TransE. Compared with our TransCat, Baseline 3 adds the description-based representation and structure-based representation together to represent the CWE entity instead of concatenating the two representations in our approach. The architecture of baseline 3 is showed in Fig. 6 In this baseline, we also use the same setting of hyperparameters as in our TransCat.

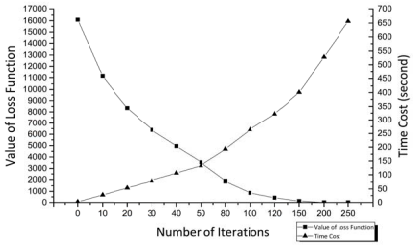


Fig. 7. The value of loss function and time cost for different numbers of iterations

C. Hyperparameters Optimization

In this section, we tune the important hyperparameters of the word embedding and TransCat model by conducting experiments with different values using the validation dataset showed in Table III. Many knowledge graph embedding based applications [5], [7], [4] report that an effective knowledge graph embedding needs well-tuned hyperparameters. In this work, since we leverage knowledge graph embeddings to learning the representation of common software weakness, we also need to consider the impact of the hyperparameters. There are four hyperparameters need to be tuned (see Table V and Fig. 7). Since the knowledge graph embedding results can support several reasoning tasks like relationship ranking, triplet classification, and multi-label classification, we select the most common task, the relation ranking task, to perform the hyperparameters optimization experiments and evaluate by Hits@1(%) which represents the proportion of correct relation type ranked in top 1. Higher *Hits@1* indicate better performance.

1) The Number of Training Iterations: Since the training objective of our approach is to minimize the margin-based score function, i.e., Eq (5), during each iteration, the number of training iterations has great impact on the performance of TransCat. The value of loss function usually decreases with the number of iterations becoming larger, which also lead to the increasing of time cost. In order to find a balance between the value of loss function and the time cost, we tune the number of training iterations with 10 increments ranged form 1 to 200. As showed in Fig. 7, we set the number of training iterations to 150, with which the the value of loss function is 150 and the time cost is 400 seconds.

2) The Batch Size for Each Training Step: In each iteration, we slice the training data into several batches based on the batch size and then train one batch in each step. Experiments are conducted to find a batch size with better performance within three commonly used values, i.e., 32, 64, 128. According to the second column in Table V, setting the batch size to 128 can obtain the highest Hits@1% on the validation dataset.

3) Margin Size: Margin is an important parameter in the margin-based loss function which affects the performance of knowledge graph embeddings greatly [19]. Since margin is nonnegative, we conduct the experiments with three values (i.e., 2.0, 5.0, 8.0) to find the suitable margin size for our approach. Based on the experiment results listed in the forth column in Table V, the margin is set to 5.0 to achieve the best performance.

4) The Dimensionality of Word Embeddings: Word embeddings are used to extract the semantic and syntactic features of words in their dimensions. Thus, the dimensionality of word embeddings is another important parameter which have impact on the performance of our approach. We conduct experiments with three values (i.e., 32, 64, 128) to optimize this parameter on the CWE textual data corpus. The last column in Table V presents the experiment results. 64-dimensional word embeddings obtain the best scores in the Hits@1%.

D. CWE Link Prediction

Link prediction tasks are among the most popular and useful applications supported by knowledge graph embeddings[5], [6], which aim at completing a triplet fact when one of the entities $((?, r, t) or (h, r, ?))$ or the relation $((h, ?, t))$ is missing. Rather than traditional prediction tasks which require only the best answer, this knowledge graph based task emphasizes more on ranking a set of candidate entities or relation types from the knowledge graph. Therefore, in this section, CWE relation ranking and CWE entity ranking experiments are conducted to test CWE and CWE-relation embeddings obtained by our TransCat model and the three baseline methods. For a given CWE triplet *(CWE head entity, relation, CWE tail entity)*,

- Relation ranking: given the *CWE head entity* and *CWE tail entity*, rank the relevant *relation* between any two CWEs among four types of relations, i.e., *parent-child*, *precede-follow*, *peerof* and *semantically related*;
- Entity ranking: given the *CWE head entity* (or *CWE tail entity*) and *relation*, rank the relevant *CWE tail entity* (or *CWE head entity*).

Two commonly used measures are considered as evaluation metrics in this task:

1) Mean Rank: represents the average ranking of correct entities or relations.

2) Hits: represents the proportion of valid entities ranked in top 10 (for entity) or top 1 (for relation), i.e., the proportion of ground truth triplets that rank in top 10 or top 1. Therefore, we prefer higher *Hits@10* and *Hits@1* and lower *Mean Rank* that indicate better performance.

RQ1: How effective is our approach in CWE relation ranking task, compared with the baseline methods?

Motivation. Our approach embeds CWE entities by concatenating description-based and structure-based representation for the relation ranking task. This is different from Baseline1 using only description-based features and Baseline2 using only structure-based features. Baseline3 uses both the description-based and structure-based representation, but different in the joint method. We want to investigate whether and to what extent our approach can improve the results of CWE relation ranking.

Approach. We perform this experiments by testing our approach and three baseline methods on the same test data, and compare the mean rank and hits@1 of different approaches.

TABLE VI
EVALUATION RESULTS ON ENTITY RANKING

Metric		Mean Rank	Hits@20(%)	Hits@40(%)	Hits@60(%)	Hits@80(%)
Baseline2	TransE (only structure)	257.3	91/387 (0.235)	118/387	139/387	156/387 (0.403)
Baseline3	TransE (structure + description)	214.2	103/387 (0.266)	133/387	154/387	169/387 (0.437)
Our Approach	TransCat (structure $\oplus$ description)	173.6	113/387 (0.292)	144/387	171/387	194/387 (0.501)

TABLE VII
EVALUATION RESULTS ON RELATION RANKING

Metric		Mean Rank	Hits@ 1(%)
Baseline1	Word Embedding (only description)	-	0.783
Baseline2	TransE (only structure)	1.685	0.623
Baseline3	TransE (structure + description)	1.426	0.762
Our Approach	TransCat (structure $\oplus$ description)	**1.294**	**0.853**

TABLE VIII
EVALUATION RESULTS ON RELATION RANKING FOR DIFFERENT TEXTUAL INFORMATION

Metric		Title	Desc.
Baseline1	Word Embedding (only description)	0.715	0.783
Baseline3	TransE (structure + description)	0.721	0.762
Our Approach	TransCat (structure $\oplus$ description)	**0.801**	**0.853**

Result. According to the results reported in Table VII, our TransCat achieves the best performance in both evaluation metrics, compared with the three baseline methods. The hits@1(%) of our approach outperforms the Baseline1, Baseline2 and Baseline3 by 7%, 23.0% and 9.1%, respectively. Similar conclusions can be drawn on mean rank except for the Baseline1 which is a classification, not ranking-based method. Our approach outperforms the Baseline2 and Baseline3 on mean rank by 0.391 and 0.132 respectively.

Our TransCat achieves the best performance on CWE relation ranking task compared with the baseline approaches. Structure-based representation is helpful in improving the results of relation ranking, but considering only structure-based representation performs worse than considering only description-based representation. Concatenation of description-based and structure-based representation outperforms addition of the two representations.

RQ2: How sensitive is our approach to the complexity of input entity descriptions/input entity representations learned from different embedding methods?

Motivation. In this experiment, we use two types of input entity textual information: the CWE titles and the CWE descriptions, which is different in the complexity of sentence. We would like to investigate whether our TransCat could still outperform the baseline method when the textual information of entity is very limited.

Approach. Since this experiment only takes text-based representation into consideration, we compare our TransCat with Baseline 1 and Baseline 3 method tested on the same test dataset. The result is evaluated by Hits@1.

Result. Table VIII presents the results of relation ranking on different textual information. We can see that our approach outperforms the Baseline1 and Baseline3 on both types of textual information. When using CWE titles as the input of learning text-based embeddings, our approach outperforms the Baseline1 and Baseline3 by 8.6% and 8%. When the CWE description is used as the input, our approach outperforms the Baseline1 and Baseline3 by 7% and 9.1%, respectively.

Our approach achieves the best performance on CWE relation ranking task by using different complexity of textual information of CWEs compared with the baseline approaches. Structure-based representation is helpful in improving the results of relation ranking, but abundant textual information could lead to better performance.

RQ3: How effective is our approach in CWE entity ranking tasks, compared with Baseline2 and Baseline3?

Motivation. Entity ranking requires reasoning over both CWE entities and relations. The Baseline1 models only CWE entities but not CWE relations, and thus is not able to fulfill the task. Therefore, this experiment compares our TransCat model with only the Baseline2 and Baseline3.

Approach. For the entity ranking task, we need to construct triplets with either head or tail CWEs removed. Regarding the strategy of constructing such triplets, we use "uniform" standards (detailed in III-E) to denote the way of replacing head or tail with equal probability [6].We apply our approach and the Baseline2 and Baseline3 methods on our test dataset, and compare the mean rank and hits for CWE entities ranked from 20 to 80 by different approaches.

Result. Table VI presents the results of ranking entities. We can see that our approach outperforms the Baseline2 and Baseline3 on mean rank by 83.7 and 40.6 respectively. And our approach performs better on all the metrics from hits@20 to hits@80.

Our approach can support knowledge graph reasoning task which is impossible for the description-only-based methods like Baseline1. Also, our approach outperforms the structure-only-based Baseline2. The difference in the joint method of description-based and structure-based representation significantly affect the performance in entity ranking tasks.

E. CWE Triple Classification

Triple classification task is to make a decision on whether the given CWE triplet *(CWE head entity, relation, CWE tail entity)* is true or false compared with the ground truth, which could be viewed as a binary classification problem.

RQ4: How effective is our approach in CWE triple classification task, compared with the Baseline2 and Baseline3?

Motivation. Similar to entity ranking task, this experiment is also only applicable to our TransCat model, the Baseline2

and Baseline3. We want to investigate whether and to what extent our approach can correctly classify the CWE triplets, compared with the Baseline2 and Baseline3 method.

Approach. Evaluation of this task needs negative labels. As CWE data does not provide negative triplets, we construct negative triplets following the same procedure as in [5], in which the head or tail CWEs are randomly replaced by another CWE (but not both at the same time) which does not have the relation (h', r, t) or (h, r, t'). Then we train a binary classifier. We set a relation-specific threshold δ_r. For a triple (h, r, t), if the value obtained by energy function $E(h, r, t) = \|h + r - t\|$ is below δ_r, the triple will be classified as positive, otherwise negative. δ_r is optimized by maximizing classification accuracies on the validation set. The training, validating and testing triplets are the same as in the CWE link prediction task (see Table III).

Result. The result is in Table IX, from which we can see that our approach outperforms the Baseline2 and Baseline3 methods on Hits@1 by 0.116 and 0.093 respectively.

Knowledge graph embedding based method can support triplet reasoning task which is impossible for the description-only-based method. Our approach slightly outperforms the structure-only-based Baseline2 and the addition of description-based and structure-based representation based Baseline3.

F. Common Consequences Prediction

In the CWE website [1], "common consequences" are listed as technical impacts that can result from each weakness (as showed in Fig. 1). In our dataset, we collect 8 types of common consequences (see Section IV-A). That is to say, if the exposure of a weakness is exploited by malicious attackers, the exploitation will result in one of the eight technical impacts or consequences[4]. Thus, we conduct experiments on predicting the common consequences using our knowledge graph embeddings, which can be regarded as a multi-label classification task [20].

RQ5: Do our approach outperforms the Baseline2 and Baseline3 method in predicting common consequences, i.e., a multi-label classification task for a given CWE?

Motivation. we want to investigate whether and to what extent our approach can perform common consequences prediction, compared with the Baseline2 and Baseline3 method.

Approach. We train a multi-label classifier which takes CWE embeddings as input and predict the labels of eight common consequences. The training and testing dataset are shown in Table IV). Since the Baseline1 is not able to perform this task, We compare our TransCat with the Baseline2 and Baseline3 methods, and evaluate the result by micro F1 and macro F1. Micro F1 calculates metrics globally by counting the total true positives, false negatives and false positives. Macro F1 calculate metrics for each label, and find their unweighted mean which does not take label imbalance into account. The two metrics are commonly used for measuring multi-label classification tasks, see [21] for details.

[4]https://cwe.mitre.org/community/swa/priority.html

TABLE IX
EVALUATION RESULTS ON TRIPLET CLASSIFICATION

Metric		Accuracy
Baseline2	TransE (only structure)	0.681
Baseline3	TransE (structure + description)	0.704
Our Approach	TransCat (structure $\oplus$ description)	**0.797**

TABLE X
EVALUATION RESULTS ON COMMON CONSEQUENCE PREDICTION

Metric		MicroF1	MacroF1
Baseline2	TransE (only structure)	0.505	0.495
Baseline3	TransAdd (structure + description)	0.588	0.574
Our Approach	TransCat (structure $\oplus$ description)	**0.627**	**0.624**

Result. The result is presented in Table X, from which we can see that the micro f1 of our approach outperforms the Baseline2 and Baseline3 mehthod by 12.2% and 3.9% respectively. The macro f1 of our approach also outperforms the Baseline2 and Baseline3 mehthod by 12.9% and 5%.

Our TransCat model can learn better CWE embeddings that better support common consequence prediction task and achieve the best performance over the baseline methods.

G. Discussion

1) Analysis of Zero-Shot Scenario: As the CWE data evolves over time, we want to make qualitative analysis on the capability of our TransCat model on predicting new triplets when at least one of the entities in triplets is out of the current knowledge graph obtained on the earlier versions of the CWE data. This situation is also known as the zero-shot scenario [18], which could not be handled by the models based on only structure-based representations such as TransE. Because the structure-based representations could not represent the entities which is not in the knowledge graph.

We use an earlier version (CWE V1.0) to simulate a zero-shot scenario. All CWE entities in CWE V1.0 are in the knowledge graph entities which can be learned through training. Changes between an earlier version (CWE v1.0) and a later version (CWE v2.0) are considered as out-of-knowledge-graph entities for the zero-shot scenario tasks. Given the head entity h and relation r embeddings learned from the CWE data V1.0, we can predict the possible tail entity t among the newly added CWEs in the V2.0. And then, we look up the newly added (h, r, t) in V2.0 to see how many predicted tail entities t on Hit@20 are matched.

Our approach can support this evolving reasoning task and it performs better in the CWEs with more changes than those with a few changes. For example, for CWE-119, there are 10 newly added relations from V1.0 to V2.0 because of the addition of new weaknesses related to CWE-119. Our approach can match 8 of the newly added relations in the "parent-child" relation on hits@20. That is, given (CWE-119, parent-child, ?) in V1.0, our approach correctly predict eight newly added CWEs in V2.0 (including CWE-786, 787, 788, 805, 825, 824,823, 822) that are related to the CWE-119. Due

to the limited number of changes on each CWE between CWE V1.0 and V2.0, our analysis of zero-shot scenario is rather anecdotal, but it does show the promising results. In the future, we will collect more pair-wise changes in the CWE evolution history to study zero-shot scenario more systematically.

2) Threats to validity: Both internal and external threats may have some impact on the validation of our experiments. Threats to internal validity relate to errors in our experimental dataset and methodology implementation. We avoid such errors by having implementation and experiment results double checked by co-authors. We have also manually checked the crawled CWEs in our dataset to ensure that they have the right relationships between each other and the common consequences labels are assigned to the right CWEs. Threats to external validity relate to the generalizability of our results. In this study, we use a medium-size training and test dataset. This allows us to perform manual analysis to understand the capability and limitations of our approach. In the future, we will reduce this threat by extending our approach to more CWES for training and testing and adapting our approach to other software weakness related databases and reasoning tasks (e.g., community detection of related CWEs).

V. Related Work

In this section, we review related work around software weakness and knowledge graph embedding.

A. Software Weakness

Since software security remains the serious challenge to the software engineering domain, there has been lots of work focusing on constructing common security repositories such as Common Vulnerability and Exposure (CVE) Database [22], Common Weakness Enumeration (CWE) Database [1], and National Vulnerability Database (NVD) [23]. Li et al. [24] design a vulnerability mining algorithm to analyze and obtain the essential characteristics of software vulnerability through mining the three repositories. Han et al. [25] propose a deep learning approach to predict multi-class severity level of software vulnerability using only vulnerability description based on CVE datasets. Katrina et al. [26] organize the common types of coding errors that lead to vulnerabilities into a taxonomy to help developers recognize categories of problems and identify existing errors.

There are also some works on predicting the relationships between two software entities as we do in this paper[27], [28], [29]. For example, Xu et al. [27] adopt word embeddings and deep learning method to predict different types of relationships between question and answer posts in Stack Overflow. Their solution solves essentially a multi-class classification problem (similar to our RQ1). As discussed in our experiments, such classification-based method does not support many knowledge graph reasoning tasks such as entity ranking task, triplet classification which require the modeling of entity relations in the knowledge graph.

B. Knowledge Graph Embedding

Knowledge graph embeddings are widely used in representing multi-relational data into a continuous low-dimensional vector space. Traditionally, the knowledge graph embedding methods focus on only structure information between entities such as TransE [5], TransH [6], TransR [7], and PTransE [7]. Closer to our work, some of the knowledge graph embedding methods also take entity information as important supplement to knowledge representation learning, such as entity descriptions and entity labels [30], [31], [32], [33], [34]. With the help of entity information, the models are capable of validating a triplet even when one of the entities in a relation is not present in the knowledge graph when embeddings are learned. For example, Xie et al. [18] propose a Description-Embodied Knowledge Representation Learning (DKRL) method for representing knowledge graph by taking advantages of entity descriptions. In their method, continuous bag-of-words and deep convolutional neural models are used to encode semantics of entity descriptions, i.e., the description-based representation. The structure-based representation follows the TransE procedure. Wu et al. [33] present a sequential text-embodied knowledge representation learning method to build knowledge representations from multiple sentences. They used recurrent neural network with pooling or long short-term memory network to encode the semantic information of reference sentence of each entity and then adopt attention model to build text-based representations of entities. Xu et al. [8] encode text description of entity by three neural models, among which an attentive model is adopted to select related information as needed, and then they integrate structure-based and text-based representations into a unified architecture by a gating mechanism. Compared with these works, our current model is much simpler and is able to achieve good performance in several knowledge graph reasoning tasks.

VI. Conclusion and Future Work

In this paper, we investigate the problem of reasoning relations and common consequences of common software weaknesses via representation learning techniques. To that end, we construct a knowledge graph representation of hundreds of common software weaknesses and thousands of relations among these weaknesses, which is crawled from the CWE database (a community-curated list of common software weaknesses). We present a knowledge graph embedding method, which combine both description-based and structure-based knowledge in the knowledge graph, to embed CWEs and CWE relations in a low-dimensional vector space. Our experiments show that our knowledge graph embedding method can capture both textual and structural knowledge about common software weaknesses for effectively supporting various reasoning tasks on software weaknesses. In the future, we will investigate more advanced models for learning description-based representation, and further investigate reasoning tasks in zero-shot scenarios.

Acknowledgment

This work has partially been sponsored by the National Science Foundation of China (No. 61572349, 61272106). Xiaohong Li is the corresponding author.

REFERENCES

[1] CWE, "Common weakness enumeration," http://cwe.mitre.org/data/index.html, [Online; accessed 15-October-2017].

[2] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," 2017.

[3] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," 2017.

[4] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," pp. 3462–347, 2017.

[5] 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.

[6] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," *AAAI - Association for the Advancement of Artificial Intelligence*, 2014.

[7] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015, pp. 2181–2187.

[8] J. Xu, X. Qiu, K. Chen, and X. Huang, "Knowledge graph representation with jointly structural and textual encoding," in *Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017, pp. 1318–1324.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[11] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[12] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[13] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[14] X. Ye, H. Shen, X. Ma, R. Bunescu, and C. Liu, "From word embeddings to document similarities for improved information retrieval in software engineering," in *Proceedings of the 38th International Conference on Software Engineering*. ACM, 2016, pp. 404–415.

[15] T. Kenter and M. De Rijke, "Short text similarity with word embeddings," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 1411–1420.

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

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research*, vol. 9, pp. 249–256, 2010.

[18] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions." in *The 30th AAAI Conference on Artificial Intelligence*, 2016, pp. 2659–2665.

[19] Y. Jia, Y. Wang, H. Lin, X. Jin, and X. Cheng, "Locally adaptive translation for knowledge graph embedding," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 992–998.

[20] M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning, "Multi-instance multi-label learning for relation extraction," in *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012, pp. 455–465.

[21] R. A. Calvo and J. M. Lee, "Coping with the news: the machine learning way," in *Ausweb*, 2003.

[22] CVE, "Common vulnerability and exposure," https://cve.mitre.org/, [Online; accessed 15-October-2017].

[23] NVD, "National vulnerability database," https://nvd.nist.gov/, [Online; accessed 15-October-2017].

[24] 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 *International Conference on Advanced Cloud & Big Data*, 2017, pp. 296–301.

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

[26] K. Tsipenyuk, B. Chess, and G. Mcgraw, "Seven pernicious kingdoms: a taxonomy of software security errors," *IEEE Security & Privacy*, vol. 3, no. 6, pp. 81–84, 2005.

[27] B. Xu, D. Ye, Z. Xing, X. Xia, G. Chen, and S. Li, "Predicting semantically linkable knowledge in developer online forums via convolutional neural network," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2016, pp. 51–62.

[28] C. Chen and Z. Xing, "Mining technology landscape from stack overflow," in *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2016, pp. 14:1–14:10.

[29] C. Chen, Z. Xing, and L. Han, "Techland: Assisting technology landscape inquiries with insights from stack overflow," in *IEEE International Conference on Software Maintenance and Evolution*, 2016, pp. 356–366.

[30] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *International Conference on Neural Information Processing Systems*, 2013, pp. 926–934.

[31] H. Zhong, J. Zhang, Z. Wang, H. Wan, and Z. Chen, "Aligning knowledge and text embeddings by entity descriptions," in *Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 267–272.

[32] D. Zhang, B. Yuan, D. Wang, and R. Liu, "Joint semantic relevance learning with text data and graph knowledge," in *The Workshop on Continuous Vector Space MODELS & Their Compositionality*, 2015, pp. 32–40.

[33] J. Wu, R. Xie, Z. Liu, and M. Sun, "Knowledge representation via joint learning of sequential text and knowledge graphs," 2016.

[34] Z. Wang and J. Li, "Text-enhanced representation learning for knowledge graph," in *International Joint Conference on Artificial Intelligence*, 2016, pp. 1293–1299.