Predicting malware threat intelligence using KGs

Nidhi Rastogi, Sharmishtha Dutta, Ryan Christian, Jared Gridley, Alex Gittens, Mohammaed Zaki, Charu Aggarwal
Rensselaer Polytechnic Institute
Troy, New York, USA

Charu Aggarwal
IBM TJ Watson
New York, USA

ABSTRACT

Large amounts of threat intelligence information about malware attacks are available in disparate, typically unstructured, formats. Knowledge graphs can capture this information and its context using RDF triples represented by entities and relations. Sparse or inaccurate threat information, however, leads to challenges such as incomplete or erroneous triples. Generic information extraction (IE) models used to populate the knowledge graph cannot fully guarantee domain-specific context. This paper proposes a system to generate a Malware Knowledge Graph called MalKG, the first open-source automated knowledge graph for malware threat intelligence. MalKG dataset (MT40K[1]) contains approximately 40,000 triples generated from 27,354 unique entities and 34 relations. For ground truth, we manually curate a knowledge graph called MT3K, with 3,027 triples generated from 5,741 unique entities and 22 relations. We demonstrate the intelligence prediction of MalKG using two use cases. Predicting malware threat information using benchmark model achieves 80.4 for the hits@10 metric (predicts the top 10 options for an information class), and 0.75 for the MRR (mean reciprocal rank). We also propose an automated, contextual framework for information extraction, both manually and automatically, at the sentence level from 1,100 malware threat reports and from the common vulnerabilities and exposures (CVE) database.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

Knowledge Graphs, Security Intelligence, Contextualization, Malware.

ACM Reference Format:
Nidhi Rastogi, Sharmishtha Dutta, Ryan Christian, Jared Gridley, Alex Gittens, Mohammaed Zaki, Charu Aggarwal and Charu Aggarwal. 2021. Predicting malware threat intelligence using KGs. In *Seoul'21: ACM Conference on Computer and Communications Security (CCS), November 14–19, 2021.* ACM, New York, NY, USA, 13 pages.

[1]Anonymous GitHub link: https://github.com/malkg-researcher/MalKG

Seoul'21, November 14–19, 2021, Seoul, South Korea

1 INTRODUCTION

Malware threat intelligence informs security analysts and researchers about trending attacks and defense mechanisms. Rooted in data, threat intelligence can help analysts' analyze zero-day attacks, newly identified vulnerabilities, threats, and attack patterns to protect intellectual property and prevent intrusions from adversaries. Extracting relevant information from heterogeneous sources requires correlation, disambiguation, and structuring before it can become actionable. Due to the sheer volume and inevitable noise in threat feeds, a central goal for security threat researchers is to overcome this issue and automate threat detection and prevention with minimal expert involvement. AI-enabled technology, together with big data about threats, has the potential to deliver on this to a great extent. Increasing sources of threat information on the Internet, such as analysis reports, blogs, common vulnerabilities, and exposure databases, provide enough data to evaluate the current threat landscape and prepare our defenses against future ones. Several research institutions, security organizations, government agencies, and experts publish this as threat reports, largely in the form of unstructured text. MITRE, NIST, OASIS Open have spearheaded initiatives that make security information available in structured formats through standards like common vulnerabilities and exposures (CVE) [43] and national vulnerability database (NVD) [32], and structured threat information eXpression (STIX) [6]. Structuring and distributing the aforementioned unstructured threat reports is critical to disseminating threat intelligence; however, this approach solves only one aspect of the problem. We need to address the following challenges as well:

(1) Whereas the current taxonomy and standards provide extensive coverage of threat concepts, they miss the semantic and contextual associations from the captured data. The extracted entities and relations ignore provenance that allows referencing to their source of publication.
(2) Limitations in data extraction models for domain-specific threat reports can lead to capturing fragmentary and sometimes fallacious threat information.
(3) Referencing and linking of data sets can address the concerns around the reliability of source information. Consequently, security analysts are required to correlate, disambiguate, and structure heterogeneous formats and quality of data before it can become actionable.

In this paper, we address the first two challenges by proposing MalKG, a knowledge graph for malware threat intelligence, and defer the third challenge for future research. MalKG extends malware ontologies [33, 39] by capturing threat information in the form of RDF triples using entities and relations. Entities are interlinked descriptions of pre-defined ontology classes such as malware,

organizations, vulnerability, attack patterns, malware behavioral characteristics, and timeline of the attack. Relations connect entities and provide context for their interpretation that describes the relationship between entities. MalKG encompasses the best of both the worlds, databases (data exploration via structured queries) and graphs (analyze like a network data structure), as well as bears formal semantics used to interpret data and infer new facts. Specifically, the main contributions of this paper are:

(1) We present MalKG, the first end-to-end, automated, ontology-defined knowledge graph generation framework for malware threat intelligence. MalKG is open source and comprises 40,000 RDF triples[2] extracted from 1,100 unstructured malware threat reports and CVE vulnerability descriptions for almost one thousand malware seen since 1999 (see Section 6 for details).
(2) We predict malware threat intelligence through two use cases to demonstrate to power of our knowledge graph. For example, MalKG predicts previously unknown threat information such as vulnerabilities, associations between malware, malware author. This is because the triples are encoded into vector embeddings that can extract latent information from graphs and reveal latent malware information unknown during training the prediction model. This is a crucial goal for cybersecurity analysts because running machine learning models on vector embeddings can produce more accurate and contextual predictions for autonomous intrusion detection systems.
(3) We present a novel, challenging corpus of extracted 27,354 unique named entities and 34 relations. This is a new benchmark framework for novel contextual security research problems.

Previous work in malware threat intelligence leans on syntactic or semantic similarities, API call sequences to detect malware or their variants [2, 4, 11, 21]. To the best of our knowledge, our research is the first that utilizes semantic similarities to predict malware features using an unstructured text corpus. We first provide two use cases that describe the predictive abilities of MalKG in Section 2 for vulnerability and malware identification. We then present MalKG's overall architecture and describe each of its component, including the corpus comprising the RDF triples, and malware feature prediction. We also discuss the challenges encountered in this process. We then discuss our evaluation and experimental results and outline ongoing and future work.

2 USE CASES

In this section, we illustrate two example scenarios to demonstrate the capabilities of MalKG in predicting malware threat intelligence.

2.1 Malware Attribution

Malware attribution is the careful process of answering the 6-Ws (who, how, what, where, why, when) of a given malware attack on a system under observation. It requires assembling sufficient malware-related features to build a fingerprint of its origination,

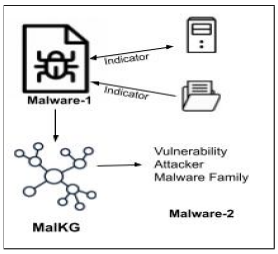


Figure 1: MalKG has captured two indicators of Malware-1. When MalKG is queried for attribution on Malware-2, it can find the associations between the two malware.

such as information about the malware author, associated campaign, and many other attributes that can reveal the malicious program's objectives or be able to recreate it. Analyzing malware and attributing it is very challenging due to the wide application of code obfuscation, polymorphism, and deformation changes in the structure of malicious code. The goal of MalKG is to automate the prediction of these features associated with a given malware (see Figure 1. For instance, for a newly discovered malware with sparse information available in the published and publicly available threat reports, MalKG can predict the attack campaigns responsible for the new malware along with a confidence score of the prediction. Threat reports[3] about the recent attacks on the SolarWinds software captured detailed analysis on the Sunspot malware that inserted the Sunburst malware into SolarWinds' Orion software. However, in the reports published before January 2021, no information was available about the threat actor or the campaign behind Sunburst. MalKG can predict malware features like attack actors and other properties that are similar to Sunburst. For instance, Sunburst and a backdoor linked to the Turla advanced persistent threat (APT) group share similarities, and MalKG can predict this information.

2.2 Vulnerability Prediction

One-fourth of the zero-day exploits detected in the year 2020 were either variants of previously publicly disclosed vulnerabilities[4] or those vulnerabilities that were incompletely patched [16]. MalKG can predict the list of potential vulnerabilities exploited by a new malware or a variant of an existing virus. For example, given a malware impacted software system (such as Microsoft Office, Linux Operating system, Windows NT Server), we are interested in predicting all the vulnerabilities (such as spoofed route pointer, multiple eval injection vulnerabilities, heap overflow) or the associated CVE IDs that may have impacted the system. MalKG can predict an ordered list of vulnerabilities or exploits based on (a) available data about the software system, (b) preliminary information about the malware, and (c) latent information captured from the knowledge graph.

[2] as of April 2021

[3] https://www.cyberscoop.com/turla-solarwinds-kaspersky/

[4] https://googleprojectzero.blogspot.com/2021/02/deja-vu-lnerability.html

3 CHALLENGES

In this section, we detail the major challenges encountered during the process of generating MalKG for making threat intelligence predictions:

(a) **Missing cybersecurity specific information extraction models** - Threat reports contain both generic and domain-specific threat data. Named entity recognizer (NER) models such as StanfordNER[5] trained on generic datasets or UMLS for specific domains such as health cannot be used for cybersecurity threat reports. Besides, using existing domain-specific concepts sometimes have titles that can mislead NER models into categorizing them as generic concepts. For example, when a CVE description mentions "Yosemite backup 8.7," a generic NER does not refer to Yosemite as a server backup software, but as a national park in California.

(b) **Missing context in extracting key phrases** - Cybersecurity standards and taxonomies overlook the importance of context in extracted concepts, which can be favorably added via linking and semantic metadata. Text data is not a mere aggregation of words. There is a meaning in the information and linking of concepts to outside sources is crucial to its effective use. For example, a human security analyst can correctly comprehend the meaning of the term "Aurora" in a threat report. It refers to the series of cyber attacks conducted by Advanced Persistent Threats (APTs) by China, first detected by Google in 2009. However, a computer or an AI model collecting this information might mistake it for the Northern Lights unless the security context is provided.

(c) **Lack of standard format of malware threat corpus** - Malware threat reports cover technical and social aspects of a malware attack in the form of unstructured text. Generating a knowledge graph requires extracting key phrases from these reports, which is not a trivial task. Unlike for a small curated benchmark dataset, we require an information extraction systems that is scalable and replicable for a large amount of text corpus specific to malware threat domain. The varying degrees of attack and defence description and the changing landscape of the information can render prior information defunct, which is an additional challenge.

(d) **Missing validation techniques** - Knowledge graphs for cybersecurity is a relatively new approach for providing structured and semantically rich information to people and machines. However, since this is an area of novel research, there was no publicly available "ground truth" to validate MalKG and measure if the prediction from MalKG were semantically correct and correspond to the so-called "real" world.

4 BACKGROUND

4.1 Malware Knowledge Graph (MalKG)

Definitions: The malware knowledge graph, MalKG, expresses heterogeneous multi-modal malware threat data as a directed graph, MalKG = $\{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{E}, \mathcal{R}$ and $\mathcal{T}$ indicate the sets of entities, relations and triples (also called facts), respectively. Each triple $\langle e_{head}, r, e_{tail} \rangle \in \mathcal{T}$ indicates that there is a relation $r \in \mathcal{R}$ between $e_{head} \in \mathcal{E}$ and $e_{tail} \in \mathcal{E}$. Knowledge graph completion, sometimes also known as entity prediction, is the task of predicting the best candidate for a missing entity. Formally, the task of entity-prediction is to predict the value for **h** given ⟨**?, r, t**⟩ or **t** given ⟨**h, r, ?**⟩, where "**?**" indicates a missing entity.

For the entities and relation e_{head}, e_{tail}, r we use the bold face **h, t, r** to indicate their low-dimensional embedding vectors, respectively. We use d_e and d_r to denote the embedding dimensions of entities and relations, respectively. The symbols n_e, n_r, and n_t denote the number of entities, relations, and triples of the KG, respectively. We use the f_ϕ notation for the scoring function of each triple ⟨**h, r, t**⟩. The scoring function measures the plausibility of a fact in $\mathcal{T}$, based on translational distance or semantic similarity [24]. A summary of the notation appears in Table 1.

Table 1: Notations and Descriptions

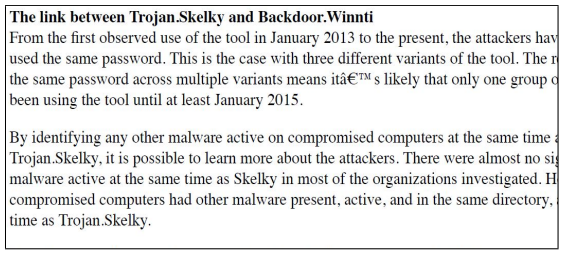
Notation	Description
$\mathcal{E}$	the set of all entities
$\mathcal{R}$	the set of all relations
$\mathcal{T}$	the set of all triples
$\langle e_{head}, r, e_{tail} \rangle$	head entity, relation and tail entity
⟨**h, r, t**⟩	embedding of a **h**, **r** and a **t**
d_e, d_r	embedding dimension of entities, relations
n_e, n_r, n_t	number of **h+t**, **r**, triples
f_ϕ	scoring function

Structure of MalKG: The MalKG comprises hierarchical entities. For example, a Malware class can have sub-classes such as TrojanHorse and Virus. A hierarchy captures the categories of malware based on how they spread and infect computers. The relations in MalKG, on the other hand, are not hierarchical. For instance, a Virus (sub-class of Malware) known as WickedRose exploits a Microsoft PowerPoint vulnerability and installs malicious code. Here "exploits" is the relation between the entities Virus (a subclass) and Software, and therefore does not require a sub-relation to describe a different category of exploit. MalKG stores facts like these in the form of a triple.

Instantiating MalKG: The malware threat intelligence knowledge graph, MalKG, is instantiated from threat reports collected from the Internet and CVE (see Figure 2). We tokenize approximately 1,100 threat reports in PDF format and from the CVE vulnerability description database to extract entities covering malware threat intelligence. The instantiating process is described in later sections and includes algorithms for named entity extraction (NER) and relation extraction (RE). Together, both NER and RE algorithms are deployed and automated to generate thousands of RDF triples that capture malware threat information into units of a knowledge graph.

MalKG Embeddings: Triples of MalKG are represented by a relatively low (e.g., 30-200) dimensional vector space embeddings [5, 41, 44, 47, 48], without losing information about the structure and preserving key features. We construct embeddings for all the triples in MalKG where each triple has three vectors, $\mathbf{e_1}, \mathbf{e_2} \in \mathbb{R}^{d_e}$, $r \in \mathbb{R}^{d_r}$ and the entities are hierarchical. Therefore, to narrow down

[5]https://nlp.stanford.edu/software/CRF-NER.shtml


The link between Trojan.Skelky and Backdoor.Winnti
From the first observed use of the tool in January 2013 to the present, the attackers hav
used the same password. This is the case with three different variants of the tool. The r
the same password across multiple variants means itâ€™s likely that only one group o
been using the tool until at least January 2015.

By identifying any other malware active on compromised computers at the same time
Trojan.Skelky, it is possible to learn more about the attackers. There were almost no sig
malware active at the same time as Skelky in most of the organizations investigated. H
compromised computers had other malware present, active, and in the same directory,
time as Trojan.Skelky.

Figure 2: An excerpt of a threat report on Backdoor.Winnti malware family.

the choice of embedding models, we ensured that the embedding model meets the following requirements:

(1) An embedding model has to be expressive enough to handle different kinds of relations (as opposed to general graphs where all edges are equivalent). It should create separate embeddings for entities that can be involved in the same relation. To elaborate, for two triples- $\langle e_1, \textit{hasVulnerability}, e_3 \rangle$ and $\langle e_2, \textit{hasVulnerability}, e_3 \rangle$, the embedding of an entity $e_1 \in$ **class:ExploitTarget** is different than that of an entity $e_2 \in$ **class:Software**.
(2) An embedding model should also be able to create different embeddings for an entity when it is involved in different kinds of relations [45]. This requirement ensures creating embedding of triples with 1-to-n, n-to-1, and n-to-n relations. For example, a straightforward and expressive model TransE [9] has flaws in dealing with 1-to-n, n-to-1, and n-to-n relations [25, 45].
(3) Entities that belong to the same class should be projected in the same neighborhood in the embedding space [14]. This requirement allows using an embedding's neighborhood information when predicting the class label of an unlabeled entity (this task is called entity classification).

4.2 Threat Intelligence Prediction

Predicting malware threat information can be modeled in MalKG by defining that unknown threat information as missing from the knowledge graph. For example, consider "Adobe Flash Player" as an entity belonging to the class Software in MalKG. A user is interested in learning all the vulnerabilities impacting "Adobe Flash Player." which are mostly unknown so far. It can be written as an incomplete triple $\langle \textit{Adobe Flash Player, hasVulnerability, ?} \rangle$. The knowledge graph completion task would predict its missing tail **th**[44], i.e., a vulnerability of the software titled "Adobe Flash Player." MalKG can predict missing entities from an incomplete triple, which forms the smallest malware threat intelligence unit. Section 7.1 elaborates on this topic.

In another example, a user is interested in learning information associated with a malware family named "Hydraq". Some information on this malware is known and therefore contained in the training set. However, some unknown information (e.g., campaign that involved this malware) may be missing from the KG. This missing information can be written as an incomplete triple $\langle \textit{?, involvesMalware, Hydraq} \rangle$. The knowledge graph completion task would predict its missing head **h**[44], i.e., a campaign that involved this Hydraq malware family. MalKG can predict missing entities from an incomplete triple, which forms the smallest malware threat intelligence unit. Section 7.1 elaborates on this topic.

5 MALKG: SYSTEM ARCHITECTURE

The input to our proposed framework for MalKG construction, as shown in Figure 3, consists of a training corpus and a malware threat ontology (e.g.,[33]) that has classes and relations to provide structure to MalKG. The training corpus contains malware threat intelligence reports gathered from the Internet, manual annotations of the entities and relations in those reports using the ontology, and the CVE vulnerability database. The training corpus and the ontology classes are fed into entity extractors such as SetExpan [34] to provide entity instances from the unstructured malware reports. A relation extraction model [49] uses these entity instances and ontology relations as input and predicts relations among the entity pairs. Entity pairs and the corresponding relations are called triples and they form an initial knowledge graph. We use a tensor factorization-based approach, TuckER[5], to create vector embeddings to predict the missing entities and obtaining the final version of MalKG with the completed RDF triples.

5.1 Generating Contextual and Structured Information

This section describes the process of generating contextual and structured malware threat information from unstructured data.

Corpus - Threat reports, blogs, threat advisories, tweets, and reverse engineered code can be investigated to gather details about malware characteristics. Our work relies on threat reports to capture information on the detailed account of malware attacks and CVE vulnerability descriptions. Security organizations publish in-depth information on how cyber-criminals abuse various software and hardware platforms and security tools. See Figure 2 for an excerpt of a threat report on the Backdoor.Winnti malware family. Information collected from a corpus of threat reports includes vulnerabilities, platforms impacted, modus operandi, attackers, the first sighting of the attack, and facts such as indicators of compromise. These reports serve as research material for other security analysts who acquire essential information to study and analyze malware samples, adversary techniques, learn about zero-day vulnerabilities exploited, and much more. However, not all reports are consistent, accurate, or comprehensive and can vary in technical breadth of the malware attack description.

Contextualizing using Linked Data - Each entity in MalKG has a unique identifier, which captures the provenance information comprising references to the triples' threat sources. For MalKG, this provenance information comes from two sources, a URL of the threat reports from which it was extracted, the unique CVE IDs for vulnerabilities. For general concepts, entities are linked to DBPedia [3].

Generating Triples - MalKG captures threat information in a triple, which is the smallest unit of information in the knowledge

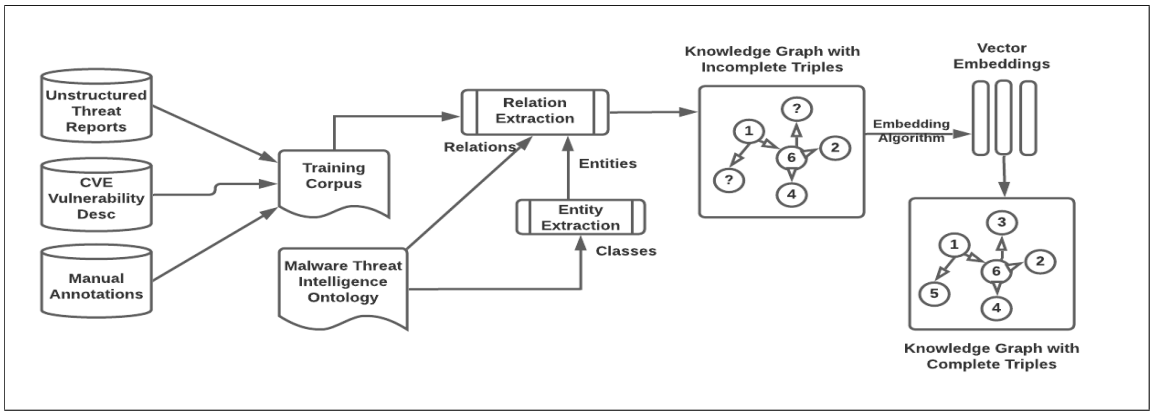


Figure 3: System Architecture for MalKG construction and showing entity prediction.

graph [26]. For instance, consider the following snippet from a threat report:[6]

"... DUSTMAN can be considered as a new variant of ZeroCleare malware ... both DUSTMAN and ZeroCleare utilized a skeleton of the modified "Turla Driver Loader (TDL)" ... The malware executable file "dustman.exe" is not the actual wiper, however, it contains all the needed resources and drops three other files [assistant.sys, elrawdisk.sys, agent.exe] upon execution..."

Table 2 shows a set of triples generated from this text. Together, many such triples form the malware knowledge graph where the entities and relations model nodes and directed edges, respectively. Figure 4 shows the sample knowledge graph from the text. A knowledge graph contains 1-to-1, 1-to-n, n-to-1, and n-to-n relations among entities. In Figure 4, the entity "dustman.exe" is an instance of a Malware File. It drops three different files. Entities DUSTMAN and ZeroCleare both *involve* the Turla Driver Loader (TDL) file, and elaborate how n-to-1 relations exist in a knowledge graph; n-to-n works similarly.

Table 2: Triples corresponding to the threat report.

Head	Relation	Tail
⟨DUSTMAN,	similarTo,	ZeroCleare⟩
⟨DUSTMAN,	involves,	Turla Driver Loader(TDL)⟩
⟨ZeroCleare,	involves,	Turla Driver Loader(TDL)⟩
⟨DUSTMAN,	involves,	dustman.exe⟩
⟨dustman.exe,	drops,	assistant.sys⟩
⟨dustman.exe,	drops,	elrawdisk.sys⟩
⟨dustman.exe,	drops,	agent.exe⟩

Inference Engine - We extend the classes and relations defined in MALOnt [33] and malware ontology [39] to extract triples from unstructured text. While using an ontology is not the only approach to instantiate MalKG, it can map disparate data from multiple sources into a shared structure and maintains a logic in that structure using rules. An ontology uses a common vocabulary that can facilitate the collection, aggregation, and analysis of that data

[6]https://www.lastwatchdog.com/wp/wp-content/uploads/Saudi-Arabia-Dustman-report.pdf

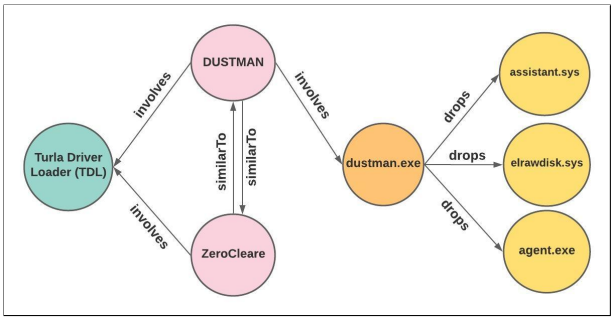


Figure 4: Sample MalKG. Nodes and edges represent entities and relations, respectively. The edge labels denote relation type.

[29]. For example, there are classes named Malware and Location in MALOnt. According to the relations specified in MALOnt, an entity of Malware class can have a relation "similarTo" with another Malware class entity. But due to the rule engine, Malware class can never be related to an entity of Location class.

Creating Ground Truth - We create ground truth triples by manually annotating threat reports. We generated MT3K, a knowledge graph for malware threat intelligence using hand annotated triples. While this was an essential step for creating ground truth, the hand-annotated approach had several limitations, but is invaluable for testing large-scale models. Details of creating the ground truth knowledge graph are covered in Section 6.2.

5.2 Predicting Missing Threat Intelligence

Knowledge graph completion enables inferring missing facts based on existing triples in MalKG when information (entities and relations) about a malware threat is sparse. For MalKG, we perform extensive experimental evaluation and predict entities through TuckER [5] and TransH [45]. Standard datasets such as FB15K [9], FB15k-237 [40], WN18 [9] , and WN18RR [12] were used to compare with other models and create baselines for other more elaborate models. However, the datasets that were used for comparison differed starkly form our MT3KG and MT40K triple dataset.

Defining MalKG protocol for Entity Prediction: The following protocol describes the prediction for missing entities in the malware knowledge graph:

a The head entity **h** is replaced with all other entities in the knowledge graph for each triple in the test set. This generates a set of candidate triples to compare with each true test triple. This step is known as corrupting the head of a test triple, and the resultant triple is called a corrupted triple.
b The scoring function, $f_\phi(h, r, t)$, returns a score for each true test triple and its corresponding corrupted triples. The scores are sorted to obtain the rank of the true test triple in the ordered set. This rank assesses the model's performance in predicting *head* given $\langle$ *?, relation, tail* $\rangle$. A higher rank indicates the model's efficiency in learning embeddings for the entities and relations.
c The previous two steps are repeated for the tail entity **t**, and true test triples ranks are stored. This rank assesses the model's performance in predicting the missing **t** given $\langle$**h, r, t**$\rangle$.
d Mean Rank, Mean Reciprocal Rank (MRR), and Hits@n are calculated from the ranks of all true test triples. Mean rank signifies the average rank of all true test triples. Mean rank (MR) is significantly affected when a single test triple has a bad rank. Therefore, recent works [5, 22] report mean reciprocal rank (MRR), which is the average of the inverse of the ranks of all the true test triples, for more robust estimation. Hits@n denotes the ratio of true test triples in the top n positions of the ranking. When comparing embedding models for prediction, smaller values of MR and larger values of MRR and Hits@n indicate a better model.

6 EXPERIMENTAL EVALUATION

This section describes our experimental setup, results, and implementation details for knowledge graph completion.

6.1 Datasets

In this paper, we extract relavant key phrases from malware threat intelligence reports and CVE dataset. The threat reports were written between 2006-2021 and all CVE vulnerability descriptions created between 1990 to 2021. The threat reports are formal investigative reports on malware attacks. Each report starts with a table of content followed by a detailed analysis of the attack patterns, actors involved, systems impacted. Images, text embedded within images, tables, and snapshots of code snippets frequently appear in these reports. Each report often focuses on the technical analysis and mitigation of one malware campaign. Below we detail further information about the threat reports:

- 1,100 threat advisory reports from all the major organizations were included in this dataset. For example, Symantec, Kaspersky, Microsoft, IBM, FireEye, TrendMicro, McAfee, and several smaller players. The reports are made available for download as well. [7]

[7]https://github.com/malkg-researcher/MalKG

Table 3: Description of Datasets

Dataset	n_e	n_r	n_t	docs	avgDeg	density
MT3K	5,741	22	3,027	81	0.5273	0.00009
MT40K	27,354	34	40,000	1,100	1.46	5.34×10^{-5}
FB15K	14,951	1345	592213	-	39.61	0.00265
WN18	40943	18	151,442	-	3.70	0.00009
FB15K-237	14,541	237	310,116	-	21.32	0.00147
WN18RR	40,943	11	93,003	-	2.27	0.00005

- 1 security researcher trained 1 graduate and 5 undergraduate student to annotate the threat reports. The annotated triples were later manually verified by the same expert.

We perform several pre-processing steps on the raw corpus data. For provenance, we provide a unique identifier (URL of threat report) to every individual entity and the ID of the CVE description to vulnerability related entities. Individual sentences were extracted from each document and assigned a sentence ID. Identification of documents and sentences also provides provenance for future work in trustworthiness, contextual entity classification, and relation extraction. We tokenize and parse the plain text sentences into words with start and end positions using the spaCy default tokenizer[8] library and save them in sequential order for each document.

6.2 Benchmark dataset

In absence of a benchmark dataset for malware threat intelligence and to train and test MalKG, we manually generate MT3K triple dataset. The MT3K dataset is a collection of 5,741 triples hand-annotated by our research team and manually validated by a security expert. 81 threat reports were annotated (see Figure 5) using the Brat annotation tool[37]. In Figure 5, "PowerPoint file" and "installs malicious code" are labeled as classes Software and Vulnerability respectively. The arrow from Software to Vulnerability denotes the semantic relationship hasVulnerability between them. The annotation followed rules defined in malware threat ontologies [33, 39]. 22 other relation types (n_r) are defined in malware threat intelligence ontology[33].

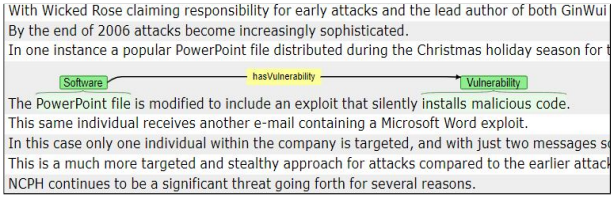


Figure 5: Annotation using Brat annotation tool.

The MT3K dataset serves as the benchmark dataset for evaluating malware threat intelligence using our automated approach for triples generation. Other triple datasets such as the FB15K [9], FB15k-237 [40], WN18 [9] , and WN18RR [12] have been used to

[8]https://spacy.io/api/tokenizer

evaluate the performance of generic knowledge graphs and their applications, such as link-prediction, and knowledge graph completion. However, there is no open-source knowledge graph for malware threat intelligence, and therefore there do no exist triples for malware KG evaluation. We hope that MT3K and MT40K will serve as benchmark datasets for future security research.

6.3 Automated Dataset: MT40K

The MT40K dataset is a collection of 40,000 triples generated from 27,354 unique entities and 34 relations. The corpus consists of approximately 1,100 de-identified plain text threat reports written between 2006-2021 and all CVE vulnerability descriptions created between 1990 to 2021. The annotated keyphrases were classified into entities derived from semantic categories defined in malware threat ontologies [33, 39]. The inference rule engine was applied when forming relations between entities to produce triples used to build the MalKG.

Our information extraction framework comprises entity and relation extraction and is grounded in the cybersecurity domain requirements. The framework takes as input the plain text of threat reports and raw CVE vulnerability descriptions. It locates and classifies atomic elements (mentions) in the text belonging to pre-defined categories such as Attacker, Vulnerability, and so on. We split the tasks of entity extraction and recognition. The entity extraction (EE) step combines an ensemble of state-of-the-art entity extraction techniques. The entity classification (EC) step addresses the disambiguation of an entity classified into more than one semantic category, and assigns only one category based on an empirically learned algorithm. Results obtained from both EE and EC are combined to give the final extraction results. Details are provided in the next section.

6.3.1 Entity Extraction (EE). We segregate the extraction of tokenized words into two categories: domain-specific key-phrase extraction and generic key-phrase extraction (class:Location). The cybersecurity domain-specific EE task was further subdivided into factual key-phrase extraction (class:Malware) and contextual key-phrase extraction (class:Vulnerability). We use precision and recall measures as evaluation criteria to narrow down the EE models for different classes. Specifically, the Flair framework [1] gave high precision scores (0.88-0.98) for classification of generic and factual domain-specific key-phrases such as - Person, Date, Geo-Political Entity, Product (Software, Hardware). The diversity in writing style for the threat reports and the noise introduced during reports conversion to text led to different precision scores for classes but within the range of 0.9-0.99. For contextual entity extraction, we expanded the SetExpan [34] model to generate all entities in the text corpus. The feedback loop that input seed values of entities based on confidence score allowed for a better training dataset. Some of the semantic classes used in this approach are Malware, Malware Campaign, Attacker Organization, Vulnerability.

6.3.2 Entity Classification (EC). A shortcoming of using multiple EE models is that some entities can get classified multiple times, which might even be inaccurate in some cases. We observed empirically that this phenomenon was not limited to any specific class. An explanation for this phenomenon is the variation and grammatical inconsistency in different threat reports and their writing style. For example, the entity "linux" was classified as Software and Organization for the same position in a document. Since the entity extraction models are running independently of each other, it was likely for close class types to have overlaps. We followed a three step approach for resolving the disambiguation in entity classification:

(1) We compared the confidence score of entities occurring at the same position in the same document and between classes.
(2) For the same confidence scores, we marked those classes as ambiguous. We retained these instances until the relation extraction phase. At that time, we relied on the rule engine from the ontology to determine the correct class of the head and tail entity for a given relation (described in the next section).
(3) For entities that passed the first two steps, we classified them using the broader semantic class.

6.3.3 Relation Extraction. We perform relation extraction for MalKG using a distantly supervised artificial neural network model [49] pre-trained on the Wiki data and Wikipedia, originally with three features. The MT3K triples and text corpus form the training dataset and the entities generated in the previous steps, and the processed text corpus the testing dataset. As described in section 3, while training and testing was not a straightforward implementation, we generated 40,000 triples using 34 relations between 27,354 entities. The instantiated entities, their classification, and position in the threat report or CVE description formed the RE model's input. The testing batch size was 16 documents. Due to scalability issues in DocRED, the number of entities per document was limited to less than 80. Text documents with a higher number of entities were further split into smaller sizes. The training ran for 4-6 hours, for an epoch size of 984, and the threshold set to 0.5.

6.4 Predicting Missing Information

Knowledge graph completion (or entity prediction) predicts missing information in MalKG. It also evaluates the entity prediction accuracy of a given triple and recommends a ranking of predicted entities. For the missing entities in the MalKG triples, a tensor factorization-based approach [5] can predict the missing entities.

6.4.1 Comparing with Benchmark Dataset. Due to the lack of a benchmark dataset in the cybersecurity domain, we rely on other datasets to observe and compare existing embedding models' performances. The most commonly used benchmark datasets are FB15K, WN18, FB15K-237, WN18RR. Collected from the freebase knowledge graph, FB15K and FB15K-237 are collaborative knowledge bases containing general facts such as famous people, locations, events. WN18 and WN18RR are datasets from Wordnet, a lexical database of the English language. Simpler models perform better on these datasets where a test triple's corresponding inverse relation exists in the training set. Such instances are eliminated from FB15K and WN18 to generate FB15K-237 and WNRR. Experimental results by OpenKE[9] for entity prediction (or knowledge graph completion) are summarized in Table 5.

[9] https://github.com/thunlp/OpenKE

The comparison between the structure of the graphs we generated - MalKG and hand-annotated, their properties, and corpus details are described in Table 3. For example, it compares the number of entities, relations, and triples (facts), the average degree of entities, and graph density of our datasets (MT3K, MT40K) and benchmark datasets. Note, the average degree across all relations and graph density are defined as n_t/n_e and n_t/n_e^2, respectively [30]. A lower score of these two metrics indicates a sparser graph.

6.4.2 Validation. MT3K validation set is used for tuning the hyperparameters for MT40K. For the TuckER experiment on MT3K, we achieved the best performance with $d_e = 200$ and $d_r = 30$, learning rate = 0.0005, batch size of 128 and 500 iterations. For the TransH experiment, $d_e = d_r = 200$, learning rate = 0.001 gave us the best results. For MT3K and MT40K, we split each dataset into 70% train, 15% validation, and 15% test data. Due to limited hand-annotated dataset, we tested the MT40K dataset using Mt3K.

7 RESULTS AND ANALYSIS

7.1 Threat Predictions

We present detailed results and analysis of two malware threat predictions that demonstrate new security findings and detect previously unknown malware-related information to the MalKG. For this, we use queries to acquire security findings from the training model. Each query (formed as an incomplete test triple) results in a set of predictions where each prediction has a confidence score associated with it. The confidence score of a predicted entity e_i denotes the probability of e_i being the accurate candidate for the incomplete triple. The predictions are sorted in descending order by this score, and we observe the top 10 predictions as potential predictions.

(1) **Case Study 1, Predicting a malware family:** A security analyst wants to identify which malware family is associated with the indicator $intel - update[.]com$ in a MalKG. The following query is posed to the MalKG in the form of an incomplete triple:

$$\langle intel - update[.]com, indicates, ?\rangle$$

Here, the accurate malware family associated with the domain $intel - update[.]com$ is named Stealer. We refer the reader to the FireEye report titled 'Operation Saffron Rose'[10] for validating this fact. The training set had some triples involving Stealer, such as:

$$\langle office.windowsessentials[.]tk, indicates, Stealer\rangle$$
$$\langle Saffron\_Rose, involvesMalware, Stealer\rangle$$

The model learns latent features of Stealer, its indicators of compromise, and other related entities (e.g., campaigns that involved Stealer) present in the training set. Table 4 shows the ordered set of predictions returned by the model, and the true answer is ranked #2. This is a significant result as it can simplify the job of an analyst to a great extent by narrowing down the scope for further analysis. Starting with this set of 10 possible choices, the analyst finds the accurate information at the second item. By definition, the Hits@3 and Hits@10 metrics increase as the true entity exist in the top 3 and certainly in the top 10. As seen in table 7, we can say 75.9% times the true entity will appear in the top 3 predictions of the model.

(2) **Case study 2, Predicting a Campaign:** A security analyst wants to identify which campaign is involved with the malware family named Hydraq. The following query is posed to the MalKG in the form of an incomplete triple:

$$\langle ?, involvesMalware, Hydraq\rangle$$

The training set had some triples involving Hydraq such as -

$$\langle Operation\_Aurora, usesMalware, Hydraq\rangle$$
$$\langle Operation\_Aurora, targets, Google\rangle$$
$$\langle Google\_Hack\_Attack, targets, Google\rangle$$

Note that the training set contains facts from different reports, and the terms 'Google Hack Attack' and 'Operation Aurora' refer to the same campaign. The model learns latent features of these entities and relations during training and later uses these features to predict a set of campaign names highly probable to involve the Hydraq malware family.
Table 4 demonstrates the predictions for this query. We can see that the true answer (Google Hack Attack) is ranked #7. In order to validate this finding, we refer the reader to the threat report titled "In-depth Analysis of Hydraq"[11].
This result contributes to increasing only the Hits@10 metric (since the rank is within the top 10), and it does not contribute to Hits@1 and Hits@3. We obtain an intuition about what the Hits@n metric on the overall test dataset represents. Hits@10 score reported in table 7 can be interpreted that 80.4% time the true entity would appear in the top 10 predictions. We elaborate on this topic in the following section.

7.2 Analysing MalKG predictions

Table 5 displays the results of entity prediction existing embedding models on the benchmark datasets. An upward arrow (↑) next to an evaluation metric indicates that a larger value is preferred for that metric and vice versa. For ease of interpretation, we present Hits@n measures as a percentage value instead of a ratio between 0 and 1. TransE has the largest Hits@10 and MRR scores on the FB15K-237 dataset, whereas TransR performs better on the rest of the three datasets. However, TransH is more expressive (capable of modeling more relations) than TransE despite having the simplicity of TransE (time complexity $\mathcal{O}(d_e)$). The space complexity of TransR ($\mathcal{O}(n_e d_e + n_r d_e d_r)$) is more expensive than TransH ($\mathcal{O}(n_e d_e + n_r d_e)$)[44]. Additionally, on average, it takes six times more time to train TransR[15] compared to TransH[25]. Considering these model capabilities and complexity trade-offs, we choose TransH as a representational translation-based model. DistMult performs better than TuckER by a small margin of 0.37% in only

[10] https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-operation-saffron-rose.pdf

[11] https://paper.seebug.org/papers/APT/APT_CyberCriminal_Campagin/2010/in-depth_analysis_of_hydraq_final_231538.pdf

[15] On benchmark datasets

Table 4: Detailed result of Entity prediction case studies

	Case study 1: Predicting a malware family $\langle intel - update[.]com, indicates, ?\rangle$		Case study 2: Predicting a Campaign $\langle ?, involvesMalware, Hydraq\rangle$	
Rank	Predicted entity	Confidence score	Predicted entity	Confidence score
1	A malware hash[12]	0.5769	Night Dragon	0.0085
2	**Stealer**	**0.5694**	Google	0.0084
3	A malware hash[13]	0.5679	IXESHE	0.0072
4	Gholee	0.5679	Hydraq	0.0053
5	200.63.46.33	0.5668	MiniDuke	0.0053
6	26978_ns2._aeroconf2014[.]org	0.5662	Poison Ivy	0.0052
7	A malware hash[14]	0.5620	**Google Hack Attack**	**0.0052**
8	Capstone Turbine	0.5602	IRC	0.0044
9	2012/06/06	0.5592	Pitty Tiger	0.0043
10	Google Hack Attack	0.5566	Miniduke	0.0039

Bold text denotes the true entity for the corresponding test triple

Table 5: Entity Prediction for different Data sets

	FB15K		WN18		FB15K-237		WN18-RR	
Model	Hits@10↑	MRR↑	Hits@10↑	MRR↑	Hits@10↑	MRR↑	Hits@10↑	MRR↑
TransE	44.3	0.227	75.4	0.395	**32.2**	**0.169**	47.2	0.176
TransH	45.5	0.177	76.2	0.434	30.9	0.157	46.9	0.178
TransR	**48.8**	**0.236**	**77.8**	**0.441**	31.4	0.164	**48.1**	**0.184**
DistMult	45.1	0.206	80.9	0.531	30.3	0.151	46.2	0.264
ComplEx	43.8	0.205	**81.5**	**0.597**	29.9	0.158	46.7	**0.276**
TuckER	**51.3**	**0.260**	80.6	0.576	**35.4**	**0.197**	**46.8**	0.272

The top three rows are models from the translation-based approach, rest are from tensor factorization-based approach

Bold numbers denote the best scores in each category

one case (Hits@10 on WN18). TuckER and ComplEx both achieve very close results on multiple cases.

Therefore, we look closely at the results of the recently refined datasets (FB15K-237 and WN18RR) as our MT3K dataset shares similar properties with these two datasets (MT3K has hierarchical relations as in WN18RR and no redundant triples as in FB15K-237). Since the space complexity is very close (ComplEx: $\mathcal{O}(n_e d_e + n_r d_e)$ and TuckER: $\mathcal{O}(n_e d_e + n_r d_r)$) we choose TuckER for evaluation on MT3K, as it subsumes ComplEx [5].

We evaluate the performance of TransH [45] and TuckER [5] on our datasets. Tables 6 and 7 display the performance of TransH and TuckER on MT3K and MT40K. TransH's MRR score on MT3K improved when compared to the benchmarks. Tucker's performance improved in all measures when compared to the other benchmark results. Evidently, TuckER outperforms TransH by a large margin on the MT3K dataset. Four reasons underlie this result:

(1) MT3K has only 22 relations in comparison to 5,741 unique entities. This makes the knowledge graph dense with a higher entities-to-relation ratio, which implies a strong correlation among entities. Tensor-factorization based approaches use tensors and non-linear transformations that capture such correlations into embeddings more accurately than linear models do[45]. For larger knowledge graphs, such as MalKG, this property is especially pertinent because it may be missing entities from the KG due to shortcomings of entity extraction models or for predicting information for a sparsely populated graph. Accurate predictions help in both scenarios, especially in the field of cybersecurity.
(2) We show that TuckER does not store all the latent knowledge into separate entity and relation embeddings. Instead, it enforces parameter sharing using its core tensor so that all entities and relations learn the common knowledge stored in the core tensor. This is called multi-task learning, which no other model except Complex[42] employs. Therefore, TuckER creates embeddings that reflect the complex relations (1-n, n-1, n-n) between entities in the MalKG.
(3) Moreover, we almost certainly ensure there are no inverse relations and redundant information in the validation and test data sets. It is hard for simple yet expressive models like TransH to perform well in such datasets compared to bi-linear models like TuckER.
(4) The number of parameters of TransH grows linearly with the number of unique entities (n_e), whereas for TuckER, it grows linearly for the embedding dimensions (d_e and d_r). A TuckER low-dimensional embedding can be learned and reused across various machine learning models and performs

[12] 01da7213940a74c292d09ebe17f1bd01
[13] a476dd10d34064514af906fc37fc12a3
[14] deeac56026f3804968348c8afa5b7aba10900aeabee05751c0fcac2b88cff71e

Table 6: Experimental evaluation of MT3K for entity prediction.

Model	Hits@1↑	Hits@3↑	Hits@10↑	MR↓	MRR↑
TransH	26.8	50.5	65.2	32.34	0.414
TuckER	64.3	70.5	81.21	7.6	0.697

Table 7: Experimental evaluation of MT40K for entity prediction.

Model	Hits@1↑	Hits@3↑	Hits@10↑	MR↓	MRR↑
TuckER	73.9	75.9	80.4	202	0.75

tasks such as classification and anomaly detection. We explore anomaly detection as future work in the MalKG to identify trustworthy triples.

Since TuckER performed significantly better than TransH on the ground truth dataset, we employ only TuckER on the MT40K dataset. It is noteworthy that all the measures except mean rank (MR) improved on MT40K compared to MT3K. It confirms the general intuition that as we provide more data, the model learns the embeddings more effectively. MR is not a stable measure and fluctuates even when a single entity is ranked poorly. We see that the MRR score improved even though MR worsened, which confirms the assumption that only a few poor predictions drastically affected the MR on MT40K. Hits@10 being 80.4 indicates that when the entity prediction model predicted an ordered set of entities, the true entity appeared in its top 10 for 80.4% times. Hits@1 measure denotes that 73.9% times, the model ranked the true entity at position 1, i.e., for 80.4% cases, the model's best prediction was the corresponding incomplete triple's missing entity. We can interpret Hits@3 similarly. We translate the overall entity prediction result on MT40K to be sufficient such that the trained model can be improved and further used to predict unseen facts.

8 RELATED WORK

Knowledge graphs store large amount of domain specific information in the form of triples using entities, and relationships between them [26]. General purpose knowledge graphs such as Freebase[8], Google's Knowledge Graph[16], and WordNet[13] have emerged in the past decade. They enhance "big-data" results with semantically structured information that is interpretable by computers, a property deemed indispensable to build more intelligent machines. No open-source malware knowledge graph exists, and therefore, in this research, we propose the first of its kind automatically generated knowledge graph for malware threat intelligence.

The MalKG generation process uses a combination of curated and automated unstructured data [26] to populate the KG. This has helped us in achieving the high-accuracy of a curated approach and the efficiency of an automated approach. MalKG was constructed using existing ontologies [33, 39] as they provide a structure to MalKG generation with achievable end goals in mind. Ontologies are constructed using competency questions that pave way for systematic identification of semantic categories (called classes) and properties (also called relations) [33]. An ontology also avoids generation of redundant triples and accelerates the triple generation process.

Several generic and specific knowledge graphs such as Freebase[8], Google's Knowledge Graph[17], and WordNet[13] are available as open source. Linked Open data[7] refers to all the data available on the Web, accessible using standards such as Uniform Resource Identifiers (URIs) to identify entities, and Resource Description Framework (RDF) to unify entities into a data model. However, it does not contain domain-specific entities that deal with cyber threats and intelligence. Other existing knowledge graphs are either not open source, too specific, or do not cater to the breadth of information our research intends to gather from intelligence reports.

Creating and completing a knowledge graph from raw text comprises multiple tasks, e.g., entity discovery, relation extraction, and designing embedding models. In this section, we review notable research works to accomplish these tasks. Entity discovery refers to categorizing entities in a text according to a predefined set of semantic categories.

Named Entity Recognition (NER) is a machine learning task for entity discovery. The machine learning model learns features of entities from an annotated training corpus and aims to identify and classify all named entities in a given text. Deep neural network-based approaches have become a popular choice in NER for their ability to identify complex latent features of text corpora without any human intervention for feature design[24]. Long Short Term Memory (LSTM), a recurrent neural network architecture, in NER can learn beyond the limited context window of a component and can learn long dependencies among words. Convolutional Neural Network (CNN) is a class of neural networks that learns character-level features like prefixes, and suffixes when applied to the task of NER. Chiu et al. [10] combine LSTM and CNN's strengths to learn word-level and character-level features. Lample et al.[23] use a layer of conditional random field over LSTM to recognize entities with multiple words. MGNER[46] uses a self-attention mechanism to determine how much context information of a component should be used, and is capable of capturing nested entities within a text.

In addition, pre-trained off-the-shelf NER tools such as StanfordCoreNLP[18], Polyglot[19], NLTK[20], and spaCy[21] are available. These tools can recognize entities representing universal concepts such as person, organization, and time. These tools categorize domain-specific concepts (e.g., malware campaign, software vulnerability in threat intelligence domain) as other or miscellaneous because the models are pre-trained on corpora created from news articles, Wikipedia articles, and tweets. While it is possible to train these models on a domain-specific corpus, a large amount of labeled training data is required.

[16] https://developers.google.com/knowledge-graph

[17] https://developers.google.com/knowledge-graph
[18] https://stanfordnlp.github.io/CoreNLP/
[19] https://polyglot.readthedocs.io/en/latest/
[20] https://www.nltk.org/
[21] https://spacy.io/api/entityrecognizer

The absence of sufficient training data in a specific domain such as malware threat intelligence poses a challenge to use the algorithms mentioned above and tools to extract technical terms of that domain. Corpus-based semantic class mining is ideal for identifying domain-specific named entities when there are very few labeled data instances. This approach requires two inputs: an unlabeled corpus and a small set of seed entities appearing in the corpus and belonging to the same class. The system then learns patterns from the seed entities and extracts more entities from the corpus with similar patterns. Shi et al. [36], and Pantel et al.[31] prepare a candidate entity set by calculating the distributional similarity with seed entities, which often introduces entity intrusion error (includes entities from a different, often more general, semantic class). Alternatively, Gupta et al.[15], and Shi et al.[35] use an iterative approach, which sometimes leads to semantic drift (shifting away from the seed semantic class as undesired entities get included during iterations). SetExpan[34] overcomes both of these shortcomings by using selective context features of the seeds and resetting the feature pool at the start of each iteration.

Following the entity discovery phase, extracting relations between entity pairs takes place. Deep neural network-based approaches eliminate the need for manual feature engineering. To extract relations for constructing a large scale knowledge graph, manual annotations to generate training data is costly and time-consuming. Subasic et al. [38] use deep learning to extract relations from plain text and later align with Wikidata[22] to select domain-specific relations. Jones et al. [20] propose a semi-supervised algorithm that starts with a seed dataset. The algorithm learns patterns from the context of the seeds and extracts similar relations. A confidence scoring method prevents the model from drifting away from pertinent relations and patterns. Ji et al.[19] propose APCNN, a sentence-level attention module, which gains improvement on prediction accuracy from using entity descriptions as an input. In applications where labeled data is scarce, distant supervision, a learning approach for automatically generating training data using heuristic matching from few labeled data instances is favored. Yao et al. [49] use distant supervision to extract features beyond the sentence level, and address document level relation extraction.

Vector embeddings address the task of knowledge representation by enabling computing on a knowledge graph. MalKG has highly structured and contextual data and can be leveraged statistically for classification, clustering, and ranking. However, the implementations of embedding vectors for the various models are very diverse. We can broadly categorize them into three categories: translation-based, neural-network-based, or trilinear-product-based [41]. For MalKG, we evaluate two embedding approaches that directly relate to our work:

(1) **Latent-distance based approach**: These models measure the distance between the latent representation of entities where relations act as translation vectors. For a triple $\langle \mathbf{h, r, t} \rangle$, they aim to create embeddings $\mathbf{h}$, $\mathbf{r}$, $\mathbf{t}$ such that the distance between $\mathbf{h}$ and $\mathbf{t}$ is low if $\langle h, r, t \rangle$ is a correct fact (or triple) and high otherwise. A correct fact is a triple where a relation exists between the head $\mathbf{h}$ and tail $\mathbf{t}$. The simplest application of this idea is:

$$f_\phi(\mathbf{h}, \mathbf{t}) = -|\mathbf{h} + \mathbf{r} - \mathbf{t}|_{1/2}$$

where $|\cdot|_{1/2}$ means either the l_1 or the l_2 norm.
TransE [9] creates embeddings such that for a correct triple $\langle h, r, t \rangle$, $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. However, it has the limitation of representing only 1-to-1 relations as it projects all relations in the same vector space. Thus, TransE cannot represent 1-to-n, n-to-1, and n-to-n relations. TransH [45] overcomes the drawback of TransE without increasing computational complexity. It uses a separate hyperplane for each kind of relation, allowing an entity to have different representations in different hyperplanes when involved in different relations. TransR [25] proposes a separate space to embed each kind of relation, making it a complex and expensive model. TransD [17] simplifies TransR by further decomposing the projection matrix into a product of two vectors. TranSparse [18] is another model that simplifies TransR by enforcing sparseness on the projection matrix. These models combined are referred to as TransX models. An important assumption in these models is that an embedding vector is the same when an entity appears as a triple's head or tail.

(2) **Tensor based approaches**: Tensor factorization-based approaches consider a knowledge graph as a tensor of order three, where the head and tail entities form the first two modes and relations form the third mode. For each observed triple $\langle e_i, r_k, e_j \rangle$, the tensor entry X_{ijk} is set to 1; and otherwise 0. The tensor is factorized by parameterizing the entities and relations as low dimensional vectors. For each entry $X_{ijk} = 1$, the corresponding triple is reconstructed so that the value returned by the scroing function on the reconstructed triple is very close to X_{ijk}, i.e., $X_{ijk} \approx f_\phi(\langle e_i, r_k, e_j \rangle)$. The model learns the entity and relation embeddings with the objective to minimize the triple reconstruction error:

$$L = \Sigma_{i,j,k}(X_{ijk} - f_\phi(e_i, r_k, e_j))^2$$

where f_ϕ is a triple scoring function.
Another model, RESCAL [28], comprises powerful bi-linear models, but may suffer from overfitting due to large number of parameters. DistMult [48] is a special case of RESCAL which fails to create embeddings for asymmetric relations. ComplEx [42] overcomes the drawback of DistMult by extending the embedding space into complex numbers. HolE [27] combines the expressive power of RESCAL with the efficiency and simplicity of DistMult. TuckER [5] is a fully expressive model based on Tucker decomposition that enables multi-task learning by storing some learned knowledge in the core tensor, allowing parameter sharing across relations. TuckER represents several tensor factorization models preceding it as its special case. The model has a linear growth of parameters as d_e and d_r increases.

(3) **Artificial neural network based models**: Neural network based models are expressive and have more complexity as they require many parameters. For medium-sized datasets, they are not appropriate as they are likely to over fit the data, which leads to below-par performance [44].

[22] https://www.wikidata.org/wiki/Wikidata:Main_Page

9 CONCLUSION

In this research, we propose a framework to generate the first malware threat intelligence knowledge graph, MalKG, that can produce structured and contextual threat intelligence from unstructured threat reports. We propose an architecture that takes as input an unstructured text corpus, malware threat ontology, entity extraction models, and relation extraction models. It creates vector embeddings of the triples that can extract latent information from graphs and find hidden information. The framework outputs a MalKG, which comprises 27,354 unique entities, 34 relations and approximately 40,000 triples in the form of $entity_{head}$-relation-$entity_{tail}$. Since this is a relatively new line of research, we demonstrate how a knowledge graph can accurately predict missing information from the text corpus through the entity prediction application. We motivate and empirically explore different models for encoding triples and concluded that the tensor factorization based approach resulted in the highest overall performance among our dataset.

Using MT40K as the baseline dataset, we will explore robust models for entity and relation extraction for future work that can generalize to the cybersecurity domain for disparate datasets, including blogs, tweets, and multimedia. Another application is to use vector embeddings to identify possible cybersecurity attacks from the background knowledge encoded in MalKG. The current prediction model can generate a list of top options for missing information. It will be interesting to extend this capability to predicting threat intelligence about nascent attack vectors and use that information to defend against attacks that are yet to propagate widely. A fundamental issue that we want to address through this research is inaccurate triples in the knowledge graph. This draws from the limitations of entity and relation extraction models and inaccurate descriptions in the threat reports.

REFERENCES

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 54–59.

[2] Eslam Amer and Ivan Zelinka. 2020. A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence. *Computers & Security* 92 (2020), 101760.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*. Springer, 722–735.

[4] Şerif Bahtiyar, Mehmet Barış Yaman, and Can Yılmaz Altınığne. 2019. A multi-dimensional machine learning approach to predict advanced malware. *Computer networks* 160 (2019), 118–129.

[5] Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590* (2019).

[6] Sean Barnum. 2012. Standardizing cyber threat intelligence information with the Structured Threat Information eXpression (STIX). *MITRE Corporation* 11 (2012), 1–22.

[7] Florian Bauer and Martin Kaltenböck. 2011. Linked open data: The essentials. *Edition mono/monochrom, Vienna* 710 (2011).

[8] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1247–1250.

[9] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[10] Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics* 4 (2016), 357–370.

[11] Sanjeev Das, Yang Liu, Wei Zhang, and Mahintham Chandramohan. 2015. Semantics-based online malware detection: Towards efficient real-time protection against malware. *IEEE transactions on information forensics and security* 11, 2 (2015), 289–302.

[12] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[13] Christiane Fellbaum. 2010. WordNet. In *Theory and applications of ontology: computer applications*. Springer, 231–243.

[14] Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. 2015. Semantically smooth knowledge graph embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 84–94.

[15] Sonal Gupta and Christopher D Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. 98–108.

[16] Allen D Householder, Jeff Chrabaszcz, Trent Novelly, David Warren, and Jonathan M Spring. 2020. Historical Analysis of Exploit Availability Timelines. In *13th {USENIX} Workshop on Cyber Security Experimentation and Test ({CSET} 20)*.

[17] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*. 687–696.

[18] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Thirtieth AAAI conference on artificial intelligence*.

[19] Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions.

[20] Corinne L Jones, Robert A Bridges, Kelly MT Huffer, and John R Goodall. 2015. Towards a relation extraction framework for cyber-security concepts. In *Proceedings of the 10th Annual Cyber and Information Security Research Conference*. 1–4.

[21] Chanhyun Kang, Noseong Park, B Aditya Prakash, Edoardo Serra, and Venkatraman Sivili Subrahmanian. 2016. Ensemble models for data-driven prediction of malware infections. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. 583–592.

[22] Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in neural information processing systems*. 4284–4295.

[23] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* (2016).

[24] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[25] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.

[26] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proc. IEEE* 104, 1 (2015), 11–33.

[27] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2015. Holographic embeddings of knowledge graphs. *arXiv preprint arXiv:1510.04935* (2015).

[28] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data.. In *Icml*, Vol. 11. 809–816.

[29] Alessandro Oltramari, Lorrie Faith Cranor, Robert J Walls, and Patrick D McDaniel. 2014. Building an Ontology of Cyber Security.. In *Semantic Technology for Intelligence, Defense and Security*. Citeseer, 54–61.

[30] Ankur Padia, Konstantinos Kalpakis, Francis Ferraro, and Tim Finin. 2019. Knowledge graph fact prediction via knowledge-enriched tensor factorization. *Journal of Web Semantics* 59 (2019), 100497.

[31] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. 938–947.

[32] Shirley Radack and Rick Kuhn. 2011. Managing security: The security content automation protocol. *IT professional* 13, 1 (2011), 9–11.

[33] Nidhi Rastogi, Sharmishtha Dutta, Mohammed J Zaki, Alex Gittens, and Charu Aggarwal. 2020. MALOnt: An Ontology for Malware Threat Intelligence. *arXiv preprint arXiv:2006.11446* (2020).

[34] Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. 2017. Setexpan: Corpus-based set expansion via context feature selection and rank ensemble. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 288–304.

[35] Bei Shi, Zhengzhong Zhang, Le Sun, and Xianpei Han. 2014. A probabilistic co-bootstrapping method for entity set expansion. (2014).

[36] Shuming Shi, Huibin Zhang, Xiaojie Yuan, and Ji-Rong Wen. 2010. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. 993–1001.

[37] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. 102–107.

[38] Pero Subasic, Hongfeng Yin, and Xiao Lin. 2019. Building Knowledge Base through Deep Learning Relation Extraction and Wikidata.. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*.

[39] Morton Swimmer. 2008. Towards an ontology of malware classes. *Online] January* 27 (2008).

[40] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 1499–1509.

[41] Hung Nghiep Tran and Atsuhiro Takasu. 2019. Analyzing knowledge graph embedding methods from a multi-embedding interaction perspective. *arXiv preprint arXiv:1903.11406* (2019).

[42] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML).

[43] Common Vulnerabilities. 2005. Common vulnerabilities and exposures.

[44] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[45] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes.. In *AAAI*, Vol. 14. Citeseer, 1112–1119.

[46] Congying Xia, Chenwei Zhang, Tao Yang, Yaliang Li, Nan Du, Xian Wu, Wei Fan, Fenglong Ma, and S Yu Philip. 2019. Multi-grained Named Entity Recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 1430–1440.

[47] Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. 2015. TransA: An adaptive approach for knowledge graph embedding. *arXiv preprint arXiv:1509.05490* (2015).

[48] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).

[49] Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A Large-Scale Document-Level Relation Extraction Dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 764–777.