WIP: Enhancing Security Event Detection on Twitter with Graph-based Tweet Embedding

Jian Cui
Indiana University Bloomington
cuijian@iu.edu

***Abstract*—Twitter has been recognized as a highly valuable source for security practitioners, offering timely updates on breaking events and threat analyses. Current methods for automating event detection on Twitter rely on standard text embedding techniques to cluster tweets. However, these methods are not effective as standard text embeddings are not specifically designed for clustering security-related tweets. To tackle this, our paper introduces a novel method for creating custom embeddings that improve the accuracy and comprehensiveness of security event detection on Twitter. This method integrates patterns of security-related entity sharing between tweets into the embedding process, resulting in higher-quality embeddings that significantly enhance precision and coverage in identifying security events.**

I. Introduction

The ever-growing and increasing threats raise the need for security practitioners to stay current with timely and accurate information. Social networks, particularly Twitter (now referred to as X), have consistently served as an invaluable resource for security practitioners. Twitter has been substantiated as a valuable tool for security practitioners, aiding in vulnerability monitoring, threat intelligence gathering, and more, as indicated by a study conducted by TrendMicro [25]. Moreover, Twitter has played an important role in a variety of security applications, including vulnerability disclosure [19], IOC gathering [22], [16], and DDoS forecasting [29].

While the importance of extracting security events from Twitter is widely acknowledged, the sheer volume and inherent noise in human-crafted tweets pose significant challenges for security event detection on the platform.

Although many text embedding methods are available, such as Word2Vec [14] or BERT [6], directly clustering the embeddings generated by these models is unlikely to produce clusters that can effectively distinguish events. In Figure 1, three tweets relate to two events occurring on September 15th, 2022. Two tweets (T_{e_1}, T'_{e_1}) discuss the *WhatsApp 0-day bug* (event e_1), while another tweet (T_{e_2}) related to the *Microsoft Exchange vulnerability* (event e_2). Surprisingly, the BERT embedding distance between tweets T_{e_1} and T_{e_2} belonging to event e_1 is greater than that between tweets T_{e_1} and T_{e_2}. This can be attributed to the similarity in syntactic patterns and semantics, causing BERT embedding to inaccurately suggest proximity.

Workshop on AI Systems with Confidential Computing (AISCC) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-4-5
https://dx.doi.org/10.14722/aiscc.2024.23002
www.ndss-symposium.org

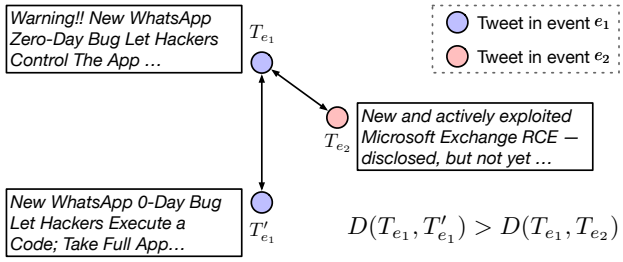

Fig. 1: Tweets embedded with BERT. The distance between embeddings of tweets belonging to the same event is larger than those belonging to different events. i.e., $D(T_{e_1}, T'_{e_1}) > D(T_{e_1}, T_{e_2})$

While previous studies [4], [7] have leveraged existing text embedding methods and showed effectiveness in event detection, they operate under the assumption that distinct events exhibit different topics, making text embeddings sufficient for effective clustering. However, in the security domain, it is notable that different events may share similar topics, such as the emergence of various ransomware strains, and exhibit similar syntactic patterns, as illustrated by the sample tweets shown in Figure 1. As such, clustering tweets is not effective when using text embeddings alone. Therefore, there is a need to create specialized methods for generating event-centric tweet representations.

Thus, in this work, we propose a novel security event detection method, *Tweezers*, that can achieve both high precision and high coverage security event detection. In order to obtain tweet embeddings that are specifically tailored for security event detection, we use a tweet relation graph and a dedicated event detection objective function. The tweet relation graph is created by connecting tweets that share the same security entities, aiming to distinguish tweets belonging to different events by capturing their different entity-sharing patterns. A Graph Attention Network (GAT) is utilized to incorporate tweet relation graphs and other features that are useful in event detection tasks, including tweet content, security categories, and temporal information. The experiment result suggests that our *Tweezers* shows a promising result in identifying new events compared to existing baselines.

To summarize, the major contributions of our work are:

- We introduce a tweet relation graph in the generation of tweet embeddings, thereby enhancing the quality of the generated embeddings.

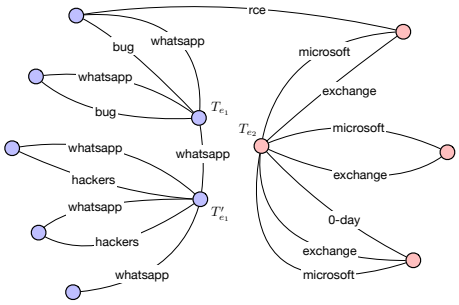

Fig. 2: In the tweet relation graph, the neighbors of tweets T_{e_1}, T'_{e_1}, and T_{e_2}—referenced in Figure 1—are illustrated. Purple and pink dots represent tweets associated with the corresponding events.

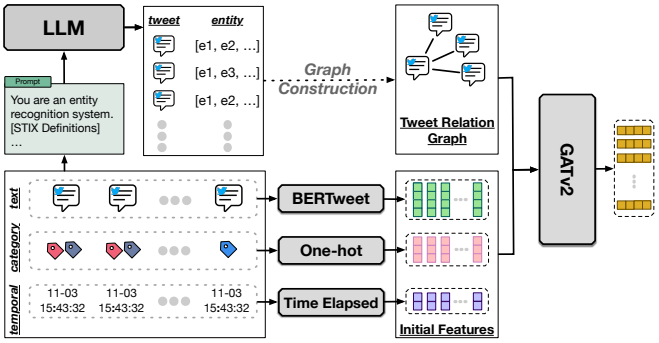

Fig. 3: Embedding generation overview

- Our proposed tweet embedding method, optimized for event detection, outperforms existing text or graph-based embedding methods.

II. Related Work

The domain of security event detection can be further categorized into two distinct categories. The first category is to detect malicious security events or activities through the analysis of system logs, syscall traces, and similar data sources. Provenance graphs, which are constructed from auditing logs, are widely used for detecting malicious behaviors [10], [11], [9], [26]. Additionally, there are also some prior works dedicated to the automated correlation of these security events [27] and the prediction of future security events [21], [15].

In another category, researchers focus on detecting emerging security events from social media, such as new data breaches, attacks, etc. Previous work employed machine learning techniques to classify tweets into predefined categories [18], [30], [33]. However, these methods primarily categorize tweets and do not identify specific event instances. Subsequently, Shin et al. [23] and Sceller et al. [13] employed keyword filtering and clustering techniques on tweet embeddings to identify specific instances of security events. These approaches, nevertheless, simply apply the standard text embedding methods on tweets, leading to less effective event identification (low event detection coverage) through clustering compared to dedicated embedding methods we proposed (shown in Section IV).

III. Graph-based Tweet Embedding Generation

A. Method Overview

To overcome the limitation of applying text embeddings, it is crucial to identify different threat-related information, such as different threat actors, victim organizations/individuals, attack patterns, activities, methods, etc. These details provide a precise description of specific aspects of a security event, so identifying tweets that share such information is crucial to distinguishing different security events. To this end, we extract the entities defined in STIX 2.1 [24] from each tweet, and to capture tweets that share the same entities, we construct a tweet relation graph, where each node represents an individual tweet and connections between nodes are established based on the shared entities. Figure 2 shows the one-hop neighbors of the tweets mentioned in Figure 1. It is worth noting that tweets belonging to event e_1 are mainly connected to other tweets in the same event, while tweets related to event e_2 are connected to other tweets in event e_2.

As such, the tweet relation graph contains entity-sharing information among tweets, serving as valuable evidence to distinguish tweets related to different events. Therefore, we incorporate this information into tweet embeddings for effective event detection. To generate tweet embeddings, we employ the Graph Attention Network (GAT), a deep learning architecture that processes graph-structure information. The GAT takes the tweet relation graph and event-related features, such as tweet content, security categories, and temporal information associated with each node, as input. GAT is trained with dedicated objective functions, ensuring the effective integration of all relevant information into the final output tweet embeddings. After training, the GAT can integrate entity-sharing information presented in the tweet relation graph and event-related features to generate effective embeddings for event detection.

B. Event-centric Embedding Generator

In the following, we provide details on the construction of the tweet relation graph and the derivation of event-related features, followed by a detailed explanation of our embedding generation methodology.

Tweet relation graph construction The tweet relation graph can be described as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the nodes (tweets), and $\mathcal{E}$ represents the edges in the graph. Tweet relation graph is constructed so that an edge connects any two tweets that share the same entities. 13 entities are defined based on the 18 STIX Domain Objects (SDOs) outlined in STIX 2.1 [24], with the exclusion of non-crucial entities for event detection, such as Report, Note, Observed data, etc.

The process of extracting these entities from unstructured text is commonly known as Named Entity Recognition (NER) in the Natural Language Processing (NLP) domain. While numerous deep learning models have been proposed and proven effective in NER tasks, they rely on human labor to annotate the predefined entities for training these NER models. However, the success of generative Language Model Models (LLMs) has simplified the NER task. As indicated

by recent studies [1], [31], prompt engineering on generative LLMs shows superior performance in NER tasks compared to traditional NER models. Aligned with the methodology proposed by previous research [1], we incorporate definitions for 13 security-related entities in the prompt, as shown in the full prompt in Figure 4 in Appendix. Subsequently, the extracted entities are used in the construction of the tweet relation graph.

The constructed Tweet Relation Graph, indicating shared entities among tweets, serves as alternative information for generating embeddings in security event detection. Also, the tweet relation graph is combined with event-related features defined in the following section, to generate more powerful embeddings with graph neural networks.

Event-related feature engineering. We identify three critical pieces of information for event detection: tweet content, security category, and temporal information. The encoding details for each piece of information are elaborated below. The resulting encoded vectors are then concatenated to form the initial features of the corresponding nodes in the tweet relation graph.

1) **Tweet content**: The (text) content of the tweet contains information indicative of the discussed event. The tweet content is converted into a 768-dimensional vector by the BERTweet model, which was chosen for its specialization in the Twitter domain. As noted previously, embeddings of tweet text alone cannot provide a comprehensive representation of a tweet's information.
2) **Security category**: Security categories can be useful to find similar events and distinguish different events. Security categories are transformed into one-hot vectors, each uniquely representing a specific category. Due to the page limit, details of how to tag security categories is not introduced in this paper.
3) **Temporal information**: Tweets describing the same event tend to be posted within a similar timeframe. The temporal information of tweets is utilized as two-dimensional features (hours and days elapsed since the beginning of 2020).

Event-centric Embedding Generation The tweet relation graph and its node features are processed with the enhanced version of GAT, GATv2 [2]. For a node (tweet) v, the output embedding $\mathbf{h}'_v$ is generated through the following equations.

$$\mathbf{h'}_v = \sum_{v' \in N(v)} \alpha \cdot \mathbf{W}\mathbf{h}_{v'} \tag{1}$$

The attention score α is obtained by:

$$\begin{aligned} e_v &= \mathbf{a}^T \cdot LeakyReLU(\mathbf{W} \cdot [\mathbf{h}_v \parallel \mathbf{h}_{v'}]) \\ \alpha &= softmax(e_v) = \frac{exp(e_v)}{\sum_{v' \in N_v} exp(e_{v'})} \end{aligned} \tag{2}$$

where $\mathbf{h}_v$ and $\mathbf{h}'_v$ are initial features of node v and its neighbor nodes v', respectively. $\mathbf{W}$ and $\mathbf{a}$ are learnable weights of the GATv2 model. Note that GATv2 utilizes attention to learn which neighboring nodes should be more influential.

To make GATv2 produce embeddings in a desired manner, an appropriate objective function must be chosen to optimize the model. Since the goal is to produce effective embeddings for clustering, contrastive optimization techniques (which have proven to be effective in many applications [32], [5]) are utilized to train the GATv2 model.

To enable clustering, embeddings of tweets belonging to the same event should be close to each other, while embeddings of tweets associated with different events should be kept far apart from each other. This can be optimized using the contrastive learning technique of triplet loss [20]. Triplet loss uses an anchor tweet embedding $\mathbf{h}_{e_i}$ and compares its distances to embeddings of a tweet of the same event and embeddings of a tweet of a different event. The formulation of triple loss in our case is:

$$\mathcal{L}_t = max(\|\mathbf{h}_{e_i} - \mathbf{h'}_{e_i} \| - \| \mathbf{h}_{e_i} - \mathbf{h}_{e_j} \| + \alpha, 0) \tag{3}$$

where $\mathbf{h}_{e_i}$ and $\mathbf{h'}_{e_i}$ refer to embeddings of two tweets belonging to the same event e_i and $\mathbf{h}_{e_j}$ refer to embeddings of a tweet belong to another event, e_j. α represents the margin between positive and negative pairs.

Another contrastive learning method is to manipulate the distance between tweets on a pairwise basis using the pairwise loss function [17]. The pairwise loss function is:

$$\mathcal{L}_p = max(\|\mathbf{h}_{e_i} - \mathbf{h'}_{e_i} \| - \| \mathbf{h}_{e_j} - \mathbf{h}_{e_k} \| + \alpha, 0) \tag{4}$$

where $\mathbf{h}_{e_i}$ and $\mathbf{h'}_{e_i}$ refer to embeddings of two tweets belonging to same event e_i, while $\mathbf{h}_{e_j}$ and $\mathbf{h}_{e_k}$ refer to embeddings of tweets belonging to separate events e_j and e_k, respectively.

Both loss functions are summed up and utilized as the objective function to optimize the learnable parameters of the GATv2 model.

To summarize, a GATv2 model is trained to take complex graph relationships and node features to generate embeddings while optimized for clustering.

IV. Evaluation

This subsection presents an evaluation comparing the performance of our proposed event-centric embedding method with existing text- and graph-based embedding methods in clustering security tweets (identifying security events).

Dataset To evaluate the clustering effectiveness of our proposed approach, we create a dataset of cybersecurity event tweets sorted into each event. We start by finding noteworthy events by monitoring three different sources: The Hacker News[1], BleepingComputer[2], and Hackmageddon[3]. For each event, we collect tweets discussing the event through a manual process of inspecting tweets from the period when the event was reported. In this study, a total of 138 events were identified between June 1, 2022, and October 21, 2022. Among these events, a total of 1,566 tweets were found to be relevant, with 764 unique users participating in the discussion of these events.

Experimental Setup: The dataset is partitioned into training, validation, and test sets based on distinct time periods, as

[1]https://www.thehackernews.com
[2]https://www.bleepingcomputer.com
[3]https://www.hackmageddon.com

TABLE I: Training, validation, test data statistics.

	# Event	# Tweet	Period
Training	91	935	2022.06.01 $\sim$ 2022.09.01
Validation	23	328	2022.09.01 $\sim$ 2022.09.25
Test	24	353	2022.09.25 $\sim$ 2022.10.21

detailed in Table I. Embeddings are generated by inputting tweets from the dataset into each embedding method, and clustering is subsequently performed using the DBSCAN clustering algorithm.

Evaluation Metric: To evaluate the clustering efficiency, we use three commonly used clustering evaluation metrics: Normalized Mutual Information (NMI), Adjusted Mutual Information (AMI), and Adjusted Rand Index (ARI).

(1) **Normalized Mutual Information (NMI):** NMI is a normalization of the Mutual Information (MI) score to scale the results between 0 and 1, with 1 indicating perfect agreement between clusters.

(2) **Adjusted Mutual Information (AMI):** AMI is a chance-corrected variant of the MI metric that accounts for the expected mutual information. AMI also ranges from 0 to 1, with 1 indicating perfect agreement.

(3) **Adjusted Rand Index (ARI):** Adjusted Rand Index is a chance-corrected variant of the Rand Index, which measures the similarity between the cluster assignments by making pairwise comparisons. ARI ranges from -1 to 1, where 1 indicates perfect agreement, 0 indicates random agreement and negative values indicate disagreement.

Baselines: We compare our methods with the following text embedding methods:

- **TF-IDF**: Term frequency-inverse document frequency (TF-IDF) is a document embedding method that utilizes frequencies of important terms in a document.
- **Word2Vec**: Word2Vec is a neural network-based model used to learn word embeddings. We use the trained model provided by spaCy[4] trained with the Skip-gram architecture.
- **PLMs**: Encoder PLMs provide contextual representations of text. The BERT, SecureBERT, and BERTweet models were tested. The uncased version of BERT was selected, which means all input texts are converted into lowercase prior to the tokenization process. The `[CLS]` token outputs produced by these models are used. BERTweet is a PLM specifically tailored to texts in the Twitter domain, and SecureBERT is a RoBERTa-based PLM adapted to the cybersecurity domain.

Following previous social event detection methods [17], [3], we also compare our embedding methods with graph-based tweet embedding methods. Prior works use general entity recognition methods implemented in spacy[5] and build graphs based on identified entities. Then, the variations of GNN, such as GCN, GAT, GraphSAGE are applied to obtain the tweet embeddings. We use the following three variations of GNNs in this evaluation.

- **GCN** [12]: GCN aggregates information from a node's direct neighbors using shared weights, capturing local structures in the graph.
- **GAT** [28]: GAT introduces attention mechanisms to node aggregation, assigning different weights to neighbors.
- **GraphSAGE** [8]: GraphSAGE samples and aggregates information from a node's neighborhood, allowing scalability.

[4]https://spacy.io

[5]https://spacy.io/api/entityrecognizer

TABLE II: Tweet Clustering Results. ↑: higher the better

	Model	AMI (↑)	ARI (↑)	NMI (↑)
Word-based	TF-IDF	0.3036	0.0552	0.5147
	Word2Vec	0.0463	0.0060	0.1135
PLMs	BERT	0.2389	0.0203	0.4395
	SecureBERT	0.3046	0.0324	0.5020
	BERTweet	0.3466	0.0676	0.5211
Graph	GCN	0.2806	0.0869	0.4455
	GAT	0.3396	0.0988	0.5274
	GraphSAGE	0.3164	0.0912	0.5019
Tweezers		**0.5919**	**0.3384**	**0.7344**

Results and Discussion: As shown in Table II, the embeddings generated by *Tweezers* demonstrate significantly better clustering performance across all three evaluation metrics compared to other baselines. The results highlight the limitation of considering only text when embedding tweets for clustering. While domain-specific models, such as SecureBERT and BERTweet, outperform general-domain PLMs, they are still not optimized to cluster tweets for event detection. Interestingly, TF-IDF shows a comparable result to PLMs. This can be attributed to the fact that our dataset only contains security event-related tweets, mitigating the issue of keyword ambiguity commonly associated with TF-IDF. Furthermore, graph-based embedding methods, designed to enhance social event detection, prove less effective compared to our approach. This inefficiency stems from their inability to identify security-related entities in tweets, resulting in graph constructed lack of information to distinguish different events for each tweets. The superior performance of *Tweezers* can be attributed to its ability to integrate the tweet relation graph and event-related features such as tweet content, security category, and temporal information.

V. Conclusion

In conclusion, our novel method for customizing text embeddings, incorporating security-related entity-sharing patterns, substantially enhances the accuracy and scope of detecting security events on Twitter. This approach overcomes the limitations of standard techniques, marking a pivotal step forward in twitter security event detection.

REFERENCES

[1] Dhananjay Ashok and Zachary C. Lipton. Promptner: Prompting for named entity recognition, 2023.

[2] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[3] Yuwei Cao, Hao Peng, Jia Wu, Yingtong Dou, Jianxin Li, and Philip S Yu. Knowledge-preserving incremental social event detection via heterogeneous gnns. In *Proceedings of the Web Conference 2021*, pages 3383–3395, 2021.

[4] Ashis Kumar Chanda. Efficacy of bert embeddings on predicting disaster from twitter data. *arXiv preprint arXiv:2108.10698*, 2021.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[7] Ali Mert Ertugrul, Burak Velioglu, and Pinar Karagoz. Word embedding based event detection on social media. In *Hybrid Artificial Intelligent Systems: 12th International Conference, HAIS 2017, La Rioja, Spain, June 21-23, 2017, Proceedings 12*, pages 3–14. Springer, 2017.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[9] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv preprint arXiv:2001.01525*, 2020.

[10] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. Nodoze: Combatting threat alert fatigue with automated provenance triage. In *network and distributed systems security symposium*, 2019.

[11] Wajih Ul Hassan, Mohammad Ali Noureddine, Pubali Datta, and Adam Bates. Omegalog: High-fidelity attack investigation via transparent multi-layer log analysis. In *Network and distributed system security symposium*, 2020.

[12] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[13] Quentin Le Sceller, ElMouatez Billah Karbab, Mourad Debbabi, and Farkhund Iqbal. Sonar: Automatic detection of cyber security events over the twitter stream. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–11, 2017.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[15] Mohammad Naseri, Yufei Han, Enrico Mariconti, Yun Shen, Gianluca Stringhini, and Emiliano De Cristofaro. Cerberus: exploring federated prediction of security events. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2337–2351, 2022.

[16] Amirreza Niakanlahiji, Lida Safarnejad, Reginald Harper, and Bei-Tseng Chu. Iocminer: Automatic extraction of indicators of compromise from twitter. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4747–4754, 2019.

[17] Jiaqian Ren, Lei Jiang, Hao Peng, Yuwei Cao, Jia Wu, Philip S Yu, and Lifang He. From known to unknown: Quality-aware self-improving graph neural network for open set social event detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1696–1705, 2022.

[18] Alan Ritter, Evan Wright, William Casey, and Tom Mitchell. Weakly supervised extraction of computer security events from twitter. In *Proceedings of the 24th international conference on world wide web*, pages 896–905, 2015.

[19] Carl Sabottke, Octavian Suciu, and Tudor Dumitraș. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 1041–1056, 2015.

[20] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[21] Yun Shen, Enrico Mariconti, Pierre Antoine Vervier, and Gianluca Stringhini. Tiresias: Predicting security events through deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 592–605, 2018.

[22] Hyejin Shin, WooChul Shim, Saebom Kim, Sol Lee, Yong Goo Kang, and Yong Ho Hwang. # twiti: Social listening for threat intelligence. In *Proceedings of the Web Conference 2021*, pages 92–104, 2021.

[23] Hyejin Shin, WooChul Shim, Jiin Moon, Jae Woo Seo, Sol Lee, and Yong Ho Hwang. Cybersecurity event detection with new and re-emerging words. In *Proceedings of the 15th ACM asia conference on computer and communications security*, pages 665–678, 2020.

[24] STIX. 18 STIX Domain Objects (SDOs). https://oasis-open.github.io/cti-documentation/stix/intro.html#:~:text=What%20is%20STIX%3F,contribute%20and%20ask%20questions%20freely.

[25] TrendMicro. Hunting Threats on Twitter: How Social Media can be used to Gather Actionable Threat Intelligence. . https://www.trendmicro.com/vinfo/es/security/news/cybercrime-and-digital-threats/hunting-threats-on-twitter, 2019.

[26] Benjamin E Ujcich, Samuel Jero, Richard Skowyra, Adam Bates, William H Sanders, and Hamed Okhravi. Causal analysis for {Software-Defined} networking attacks. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3183–3200, 2021.

[27] Thijs Van Ede, Hojjat Aghakhani, Noah Spahn, Riccardo Bortolameotti, Marco Cova, Andrea Continella, Maarten van Steen, Andreas Peter, Christopher Kruegel, and Giovanni Vigna. Deepcase: Semi-supervised contextual analysis of security events. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 522–539. IEEE, 2022.

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[29] Zhongqing Wang and Yue Zhang. Ddos event forecasting using twitter data.

[30] Semih Yagcioglu, Mehmet Saygin Seyfioglu, Begum Citamak, Batuhan Bardak, Seren Guldamlasioglu, Azmi Yuksel, and Emin Islam Tatli. Detecting cybersecurity events from noisy short text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1366–1372, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[31] Mozhi Zhang, Hang Yan, Yaqian Zhou, and Xipeng Qiu. Promptner: A prompting method for few-shot named entity recognition via k nearest neighbor search, 2023.

[32] Rui Zhang, Yangfeng Ji, Yue Zhang, and Rebecca J. Passonneau. Contrastive data and learning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*, pages 39–47, Seattle, United States, July 2022. Association for Computational Linguistics.

[33] Shi Zong, Alan Ritter, Graham Mueller, and Evan Wright. Analyzing the perceived severity of cybersecurity threats reported on social media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 1380–1390, Minneapolis, Minnesota, June 2019.

APPENDIX

Prompt

sys:
"""
You are an entity recognition system.

Defn: An entity is defined as follows:

Attack Pattern:A type of TTP that describes ways that adversaries attempt to compromise targets.
Campaign: A grouping of adversarial behaviors that describes a set of malicious activities or attacks (sometimes called waves) that occur over a period of time against a specific set of targets.
Course of Action: A recommendation from a producer of intelligence to a consumer on the actions that they might take in response to that intelligence.
Identity: Actual individuals, organizations, or groups (e.g., ACME, Inc.) as well as classes of individuals, organizations, systems, or groups (e.g., the finance sector).
Indicator: Contains a pattern that can be used to detect suspicious or malicious cyber activity.
Infrastructure: Represents a type of TTP and describes any systems, software services, and any associated physical or virtual resources intended to support some purpose (e.g., C2 servers used as part of an attack, device or server that are part of the defense, database servers targeted by an attack, etc.).
Intrusion Set: A grouped set of adversarial behaviors and resources with common properties that are believed to be orchestrated by a single organization.
Location: Represents a geographic location.
Malware: A type of TTP that represents malicious code.
Malware Analysis: The metadata and results of a particular static or dynamic analysis performed on a malware instance or family.
Threat Actor: Actual individuals, groups, or organizations believed to be operating with malicious intent.
Tool: Legitimate software that can be used by threat actors to perform attacks.
Vulnerability: A mistake in software that can be directly used by a hacker to gain access to a system or network.

Q: Given the paragraph below, identify a list of possible entities.
For each entry explain why it either is or is not an entity. Answer in the format:
1. First Candidate | True | Explanation why the word is an entity (entity_type)
2. Second Candidate | False | Explanation why the word is not an entity (entity_type)
"""

user:
"""
Paragraph:
[TWEET CONTENT COMES HERE]

Answer:
"""

Fig. 4: Prompt used for extracting STIX entities from tweets.