Sourcing Language Models and Text Information for Inferring Cyber Threat, Vulnerability and Mitigation Relationships

Erik Hemberg
MIT CSAIL
Cambridge, USA
hembergerik@csail.mit.edu

Ashwin Srinivasan
MIT CSAIL
Cambridge, USA
ashwins@alumn.mit.edu

Nick Rutar
Peraton Labs
Basking Ridge, USA
nrutar@peratonlabs.com

Una-May O'Reilly
MIT CSAIL
Cambridge, USA
unamay@csail.mit.edu

ABSTRACT

We inquire into what cyber threat, vulnerability and mitigation text sources best support inference of relational links between entries in the text sources. Our goal is to use the free-form text that conventionally has to be read by cyberhunters and analysts, to infer plausible undetected relationships between threat and defensive information. To represent the meaning of the free-text in our data sources, we use natural language processing techniques to derive text embeddings. We feed the text embeddings to a classifier that predicts the presence/absence of a relational link. We find that it is advantageous to use information that is neither too narrowly or broadly associated with the relational link. Also in this paper, we evaluate language embedding models ranging from simple to sophisticated. For our text, there is no dominant model in terms of value to the overall performance.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Artificial intelligence**; **Machine learning**; **Machine learning approaches**;

KEYWORDS

cyber security, threat hunting, Machine Learning, prediction, language model

ACM Reference Format:
Erik Hemberg, Ashwin Srinivasan, Nick Rutar, and Una-May O'Reilly. 2022. Sourcing Language Models and Text Information for Inferring Cyber Threat, Vulnerability and Mitigation Relationships. In *ACM KDD AI4Cyber: The 2nd Workshop on Artificial Intelligence- enabled Cybersecurity Analytics at KDD'22, August XX–XX, 2022, Virtual.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

KDD '22, August XX-XX, 2022, Virtual Event, WHERE

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Cyberhunters and analysts typically scan descriptions of threat behavior, vulnerabilities, and defensive measures to assess and improve their security posture. They can follow a network of links between related entries in different sources in order to find the threats in one source that exploit vulnerabilities within another source, or the mitigations in yet another source that fix the vulnerabilities, etc [14].

For example, Log4j is a popular library for logging events in Java applications. In late December, 2021 a Log4Shell vulnerability was discovered. To learn about the vulnerability, one can read its entry in the CVE database [23] as `CVE-2021-44228: Apache Log4j2 2.0-beta9 through 2.15.0`. Then, to find out what APTs might use this vulnerability and how they would do so, one needs to figure out which behavioral tactics, techniques, or procedures (TTPs) may exploit it. This information is not directly available. While the ATT&CK matrix [21] organizes techniques and procedures by tactics, there is no direct link between CVE and ATT&CK. Instead, proceeding indirectly, a cyberhunter would have to follow the weakness links from the CVE entry, then follow all their links to CAPEC attack patterns [22], and finally look for any links to the ATT&CK matrix from the attack patterns. One possible path is `CWE-20: Improper Input Validation` to `CAPEC-136: LDAP Injection` to `T1203: Exploitation for Client Execution` and privilege escalation, `T1055: Process Injection`.

In general, the complicated nature of this sort of manual process is exacerbated by two additional challenges. Links between entries in different sources are relatively infrequent given the number of entries in each source, and links between entries only reveal documented rather than plausible, but as yet undetected relationships.

Our goal is to examine the names and descriptions of entries that are available and, from them, automatically infer a relational link between two entries from different sources that is "missing". For inference we will train a classifier on labeled examples (no link, link) and then use it to query other unlinked pairs. This will allow us to ultimately provide a relational link to curators of the two sources so they could approve and add it between the entries, indicating it as "plausible but undetected".

We investigate relationships between public, popular and curated data sources that we extracted from BRON [13]. The relationships

we have selected (with their sources) range from threats to vulnerabilities and possible defensive mitigations and engagements, which all can play a part in the cyberhunting and analysis. These relationships are:

- *attack-pattern* **uses** *technique* (CAPEC [22], ATT&CK [21]),
- *weakness* **enables** *attack-pattern* (CWE [24], CAPEC),
- *weakness* **allows** *vulnerability* (CWE, CVE [23]),
- *countermeasure* **mitigates** *technique* (DEF3ND [18], ATT&CK),
- *engagement* **counters** *technique* (ENGAGE [25], ATT&CK)

This paper addresses two questions arising from our goal.

To start with, we can create many negative examples of entries that are not linked but we have relatively few positive examples. A trained classifier would have to rely solely on their text. However, sometimes two entries linked by a relationship are themselves linked to other entries in yet other sources, (by different relationships). This prompts **Q1**: Could information from these indirect sources improve the classifier's performance? That is, should we also provide features from these indirectly linked entries?

This question is novel and provides general insights about the nature and utility of the information within the sources. Given the specific meaning of each relationship, our findings will be specific to each relationship and to the set of sources we use. We are using *BRON* [13, 14] which is a compendium of five openly available sources used for cyberhunting. It consists of text entries from: *1)* the ATT&CK matrix [21] which describes observed threat tactics, techniques and procedures (TTPs), *2)* the CAPEC list [22] which describes attack patterns, *3)* the CWE dictionary [24] which describes weaknesses, *4)* the CVE dictionary [23] which describes vulnerabilities, *5)* the knowledge graph D3FEND[18] which describe defensive countermeasures *6)* the ENGAGE framework [25] which describe defensive engagements. By combining the sources, *BRON* provides analysts with access, through one property graph, to multiple kinds of relationships. This makes our analysis of methods for plausible link inference more robust and traversing the links across the databases conveniently seamless. *BRON* is created with the latest content of its source databases [2], making it as up to date as each data source is independently.

To answer **Q1**, from *BRON*, for each relationship, we first create a dataset with records from the two sources which have entries that are directly linked. Each record is given a label indicating presence or absence of the relationship. Then we create comparable datasets, each including some other set of sources that have records indirectly linked to the direct entries, see Figure 1 for an example of how a query of the information sources can be represented with direct and indirect relationship links from different types of data sources. This allows us to train a classifier for each dataset and compare their performances, designating the sources of dataset of the best performing model to be the most informative. Note that the primary focus of this contribution is on text information and LEM, so the classifier is fixed to provide a benchmark value. Our choice of classification method is motivated by [14] who investigated multiple classifiers for attack pattern uses technique classification and found that random forest classifiers had good performance.

In addition, we need to convert the entries composed of free-form text in each record of the datasets to a feature representation for the classifier. For this we use Language Embedding Models (LEM) to translate the text into a numeric input representation that preserves its meaning and allows use by models. A LEM is trained on a corpus of text to infer an embedding space where language meanings that are similar have similar embeddings. For example, the ATT&CK techniques of "Password Guessing" and "Password Brute Forcing" would be close in embedding space, and both further from "Password Policy Discovery". A LEM outputs a feature for the text in a high-dimensional, continuous-valued vector representation.

Table 1: Comparison of LEMs properties.

LEM	Word Context	Word Meaning	Embedding Dim.	Training
BoW	No	No	Varying	None
GloVe	No	Yes	Fixed	Pre
BERT	Yes	Yes	Fixed	Pre
FBERT	Yes	Yes	Fixed	Tune

Simple language models such as Bag-of-Words, embedding models such as word2vec ([12, 14, 26, 30]) and more complex ones such as transformers, e.g. BERT, have been used successfully in security applications [3]. BERT comes trained on a general corpus and and can also be fine-tuned with text from its user's problem domain. We summarize and contrast relevant properties of these LEMs regarding word context, meaning, embedding dimension and training in Table 1. Because options vary in LEM sophistication, our second question, **Q2**, is: Which of three different, common, off-the-shelf LEMs: Bag-of-Words (BoW), GloVe [16, 29], BERT [7], plus a fourth, fine-tuned version of BERT (FBERT), best supports ML-based inference of a missing relationship?

To determine missing relationships, see Figure 2, after choosing both a relationship we want to infer and a dataset, we use one of the four LEMs to obtain features for each record. We then use a portion of the dataset for training a classifier that predicts the label. With held out test data, we can assess and compare the accuracy of classifiers and, by implication, the LEMs to obtain an answer to **Q2**. Note that after testing, we can also present the classifier with an unlabeled record for which we are uncertain of the relationship being present, and collect records where the classifier infers a plausible link. These collected records can then be validated with the oversight of experts and expert-supported ones can be proposed to curators. We explain this in another publication [15].

With necessary brevity, we next describe experiments and results (Sec. 2), summarize related work (Sec. 3), and conclude with our contributions and future work (Sec. 4).

2 EXPERIMENTS

In this section we present the experimental setup, results and discussion, as well as limitations[1].

2.1 Setup

The three off the shelf LEMs we use are documented in Sec. A.1 of the Appendix. For the fourth, FBERT, we finetuned BERT on *BRON* text data, including CWEs, CAPECs, techniques, tactics, mitigations, and detections, using the masked language modeling objective. A 90/10 train-validation split was chosen so that the quantity of text available for finetuning was maximized, while also supporting a

[1]Source code is available at https://github.com/ALFA-group/AI4CyberMLHat_2022.

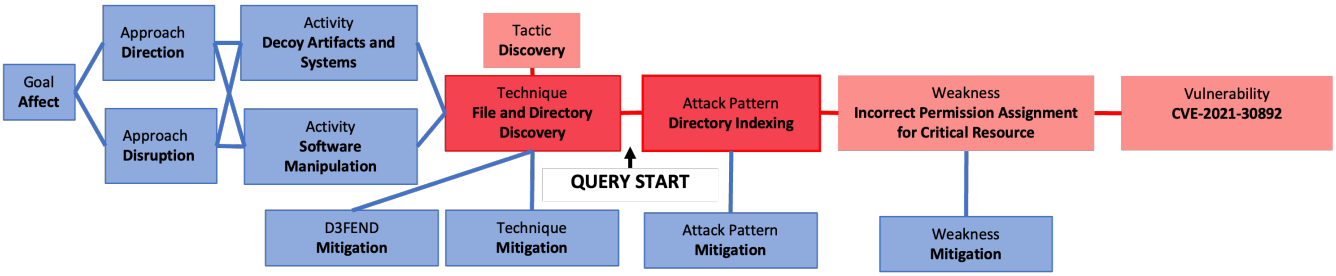


Figure 1: Query example of information sources. Dark red is Direct neighbor (N), light red is Offensive (O) and blue is Mitigation (M). This example is called a NOM dataset

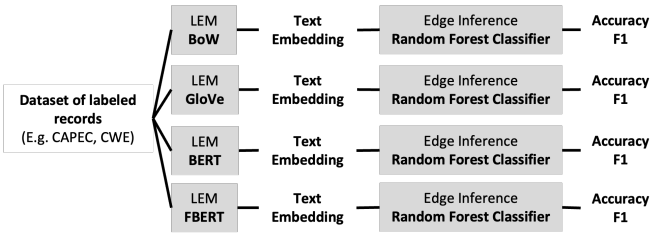


Figure 2: Flowchart of methodology. Two or more sources are selected to create exemplars for a dataset. Only one LEM is selected at a time. For each LEM a classifier is trained.

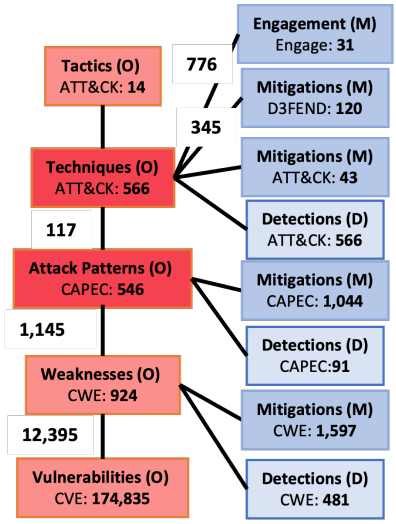


Figure 3: Visualization and statistics of *BRON*'s entries. The box identifies the description source and how many descriptions are present. Red is offensive information and blue is defensive information. Data is from 2nd Dec 2021. It also visualizes the dataset for CAPEC-Technique.

validation set to evaluate model-fit at each epoch of finetuning. Hugging Face's `DataCollatorForLanguageModeling` [32] was used to batch the training/validation data, pad the sequences to the maximum length of the batch, and randomly mask 15% of the tokens in the sequences. The model was finetuned for 50 epochs with a batch size of 8. After finetuning, for our experiments we used the model with the lowest validation loss across all epochs.

Our experimental workflow for one relationship link is as follows:

Step 1. Prepare five datasets: *N, NO, NOM, NOD, NOMD.* Our minimal dataset "Neighbors", denoted by N , is drawn from the two sources with directly linked entries. It is an aggregation of positively-labeled, directly-linked entries and negatively-labeled pairs of entries (one from each source) that have no link between them. Table 3 provides, for each relationship, how many links it has in the version of *BRON* that we worked with. It also shows the quantity and ratio of potential *undetected* relationships.

We also designate sources in *BRON* as offensive (O), defensive countermeasures (D) and mitigations (M) and create datasets by following directly linked entries' other links in these databases to identify additional text entries that may help with inferring the relationship (For details see Figure 3 and Table 2). These datasets also have records with negative labels. Table 3 shows for each relationship, how many positive examples are in *BRON*. Because of computational expense, only CVEs in 2021 were considered for the weakness **allows** vulnerability relationship. There were 174,835 CVE entries in total, and 14,613 entries in 2021. For inference, 1,000 links were randomly sampled to be used as positive examples.

Step 2. Translate Text Entries in Records to Classifer Inputs. We select a dataset. For each of its text entries, we extract the text as one (concatenated) segment and tokenize it. Tokenization includes word stemming, removal of common or connecting words [16]. We select one of the LEMs and use it to translate the tokenized segments of a record into an embedding. This yields an updated dataset replacing the text with inputs suitable for machine learning.

Step 3. Train Relationship Classifier We train a classifier (link or no link) with the updated dataset. The dataset is divided with a 70/30 train-test split. We use a `Random Forest Classifier` from Scikit-learn [27]. The classifier outputs the probability of a link. To minimize false positives, we empirically tune the class error weights of the cost matrices. We correct for the class imbalance in datasets by under-sampling the negative label records (See appendix Sec. A.2). We run the trained classifier on our held out test data and evaluate its performance in terms of accuracy and F1 score.

Step 2 is repeated with BoW and three different LEMs: GloVe, BERT, FBERT. Steps 1-3 are repeated for our five selected relationships. This yields twenty models per relationship per dataset (of

Table 2: Datasets of, narrow to broader, text. The direct set, N, is shown for the CAPEC-Technique relationship, see Figure 3.

Name	Dataset	Attack Pattern	Technique	Tactic	Weakness	Mitigative	Detections
N	Direct	✓	✓				
NO	Direct and Offensive	✓	✓	✓	✓		
NOM	Direct, Offensive and Defensive	✓	✓	✓	✓	✓	
NOD	Direct, Offensive and Mitigative	✓	✓	✓	✓		✓
NOMD	All	✓	✓	✓	✓	✓	✓

Table 3: The number of positive examples, potential links, and the ratio of linked to unlinked entries.

Relationship	Positive(link)	Potential	Ratio
attack-pattern **uses** *technique*	117	308,919	0.000038
weakness **allows** *vulnerability*	12,395	13,490,017	0.000092
weakness **enables** *attack-pattern*	1,145	503,359	0.000230
countermeasure **mitigates** *technique*	345	67,575	0.000510
engagement **counters** *technique*	776	16,954	0.004600

Table 4: Best accuracy and F1 results for each relationship (on test data, showing mean values of 100 trials). F1: higher is better, range is $[0, \ldots, 1]$

Relationship	Best LEM	Acc.	F1	Dataset
countermeasure **mitigates** *technique*	GloVe	0.917	0.919	NOD
weakness **allows** *vulnerability*	GloVe	0.908	0.907	N
engagement **counters** *technique*	GloVe	0.846	0.847	NOMD
weakness **enables** *attack-pattern*	FBERT	0.836	0.836	NOM
attack-pattern **uses** *technique*	BERT	0.758	0.753	NO

which there are four) before multiple trials. We perform 100 trials of the training and testing for statistical robustness.

2.2 Results & Discussion

Q1: Do features from indirectly linked entries help inference of a plausible relationship? For a relationship, which datasets (corresponding to sources) are the more informative?

As anticipated, the sources supporting the classifiers of the highest accuracy and F1 score varies depending on the relationship, see Table 4. In fact, each relationship's best performing classifier was trained with a different dataset (in terms of sources). Text entries that the relationship directly links (dataset (D)) are sufficient for *weakness* **allows** *vulnerability*, while all datasets (NOMD) are required by the best performing classifier in the case of *engagement* **counters** *technique* inference. Sources with additional Offensive information (NO) yielded the best performing classifier by *attack-pattern* **uses** *technique* while sources with offensive and defensive (NOD) information yielded the best by *countermeasure* **mitigates** *technique* . Sources with offensive and mitigative (NOM) information are required for the best performing classifier in *weakness* **enables** *attack-pattern*.

Table 5 shows how each dataset, for each relationship, ranked among the 5 `RandomForestClassifiers`, based on the best accuracy, with mean accuracy used as a tie-breaker, as well as the LEM that was used. The last row sums the ranks over the relationships. We also see that each dataset is best at one relationship. However, direct neighbor (N) and direct neighbor and offensive (NO) are ranked quite low on the other relations. Direct Neighbor, offensive and mitigative information (NOM) is always ranked in the top 3

Table 5: Rank of each dataset per relationship. LEM by color: gray: BoW, pink: GloVe, lime: BERT, yellow: FBERT.

	Dataset				
Relationship	N	NO	NOM	NOD	NOMD
attack-pattern **uses** *technique*	3	1	2	4	5
weakness **enables** *attack-pattern*	5	4	1	3	2
engagement **counters** *technique*	5	4	3	1	2
weakness **allows** *vulnerability*	1	5	3	2	4
countermeasure **mitigates** *technique*	5	4	2	3	1
Sum of Rank (column)	19	18	**11**	13	14

Table 6: For each relationship the sources used by the best performing classifier (Accuracy) for each LEM.

Relationship	LEM	Dataset
attack-pattern **uses** *technique*	GloVe	NOM
	BoW	NOMD
	BERT	NO
	FBERT	NO
weakness **enables** *attack-pattern*	GloVe	NOM
	BoW	N
	BERT	NOM
	FBERT	NOM
engagement **counters** *technique*	GloVe	NOMD
	BoW	NO
	BERT	NOMD
	FBERT	NOMD
weakness **allows** *vulnerability*	GloVE	N
weakness **allows** *vulnerability*	GloVe	N
	BoW	NOMD
	BERT	NOM
	FBERT	NOM
countermeasure **mitigates** *technique*	GloVe	NOD
	BoW	NOM
	BERT	NOM
	FBERT	NO

of 5. This could be interpreted as neither too much nor too little information is ideal, and that offensive and mitigation information are part of the (medium-sized) ideal set of information.

Table 6 shows the best LEM for each relationship and the dataset used. Only 10% of LEMs have the best accuracy without offensive information, 60% use mitigation and 30% use detection. For more detail see Figure 4 in the Appendix.

Q2. LEM comparison. For each relationship, Table 4 shows the LEM which results in best-models of the highest accuracy and F1 score. This reveals which LEM is featured most across the relationships. It is GloVe which features in three of the five relationships. To investigate further we checked the statistical significance, with a Wilcoxon signed-rank test with α 0.05 and Bonferroni correction, these tests show: (1) *countermeasure* **mitigates** *technique* : GloVe is significantly better than the other LEMs. (2) *weakness* **allows** *vulnerability*: GloVe is significantly better than the other LEMs. (3) *attack-pattern* **uses** *technique*: The pre-trained BERT model is significantly better than the other LEMs. (4) *weakness* **enables** *attack-pattern*: Finetuned BERT is not significantly different from

GloVe. (5) *engagement* **counters** *technique*: GloVe not significantly different from FBERT or BERT.

GloVe and BERT performed the best overall on relationship inference, with GloVe performing best on *countermeasure* **mitigates** *technique* and *weakness* **allows** *vulnerability*, BERT performing the best in *attack-pattern* **uses** *technique*, and BERT and GloVe performing similarly in *weakness* **enables** *attack-pattern* and *engagement* **counters** *technique*. BoW was never the best LEM.

Attack-pattern **uses** *technique* link inference had both the lowest accuracies and the largest variance in accuracies compared to other relationships. This may have been due to the relatively small number of examples compared to the other prediction experiments, and the relatively small compared to unlinked entries. This highlights the importance of the amount of data available.

We see that the transformer based models, BERT and FBERT always performed best with at least direct neighbor and offensive information (NO). For *engagement* **counters** *technique*, the LEM often performed well with all the possible information; direct neighbor, offensive, mitigative, and defensive (NOMD). Direct neighbor information (N) only has high accuracy for *weakness* **allows** *vulnerability*, which warrants further investigation.

Finetuning BERT, i.e., FBERT, improved performance only in *countermeasure* **mitigates** *technique* and *weakness* **enables** *attack-pattern* inferences. In fact, FBERT performed worse than pre-trained BERT in *attack-pattern* **uses** *technique* prediction. This may have been due to the fact that a very limited number of examples were available for *attack-pattern* **uses** *technique* prediction compared to the other relationship predictions. Perhaps FBERT improved performance in*weakness* **enables** *attack-pattern* inference because of the relatively higher number of training examples compared to the other predictions (2, 290 examples).

2.3 Discussion

The unique characteristics of each LEM: for example, BERT and FBERT are contextual embeddings with FBERT receiving fine-turning, whereas GloVe is a static embedding lead to mixed conclusions. This is consistent with existing cybersecurity literature where each type of LEM has been used. For example, in comparative evaluation of NLP-based approaches for linking CAPEC attack patterns from CVE vulnerability information, TF-IDF is the most suitable overall CAPEC-CVE [19]. However, for CVE to ATT&CK tactic fine-tuning RoBERTa has the best performance [3].

In text based classification a key component for the performance of the LEM is of course the data. We know that the data sources have different number of tokens, with CVEs the highest average (59.2) and tactics (2.1) the lowest. In addition, some data sources where created with an explicit intention of linking to other data sources, e.g. D3FEND and ATT&CK.

Moreover, CVE to CWE matching is enabled by keyword search according to web-site [24]. This could suggest why BoW works quite well, i.e efforts to keyword match the entries may introduce a (helpful) bias.

Limitations. Our method is general in the respect that different LEMs and classifiers can be substituted but there are several limitations in our existing implementation that may have had an impact on our results. One is that the datasets were relatively small in size. Negative examples had to be under-sampled, so there may have been bias in the particular set of examples. To mitigate this limitation it is possible to use more related data by manual labeling, one-class learning or active learning. Additionally, in CWE-CVE edge inference, for the consideration of time, only CVEs in 2021 were used. For this relationship it is possible to increase the data used. In addition, more linked data exists in references in the original data sources aggregated by *BRON*, e.g to security blogs.

3 RELATED WORK

This work is most similar to [11] that infers joint entity and relations for cybersecurity concepts by training an Neural Network on BERT embeddings. In contrast, we investigate multiple relationships, sources of information from which to infer them, and different LEM options. Other similarly predictive approaches use data sources different from *BRON* and different models [4, 6, 20, 31, 33]. More generally, there are predictive approaches using machine learning (ML) in cybersecurity [1, 5, 8–10, 17].

Our end goal is similar to those of [3, 14, 26] who aim to assist cyberhunters and threat intelligence analysts by providing behavioral information. In contrast, we investigate multiple relationships and draw our text from *BRON*. We also are first, to our knowledge, to so extensively, gauge what specific sources are key to inferring any relationship (**Q1**). E.g, when inferring an **enables** link between a weakness and an attack pattern (CWE-CAPEC), we first include a set of weakness and attack pattern texts, then another experiment includes the previous set plus any techniques and tactics linked.

4 CONCLUSIONS AND FUTURE WORK

We have investigated how LEM and text information can be used to predict relation links in cyber threat, vulnerability and mitigation data. Our contributions are:

- An evaluation of BoW and three different LEMs for machine translating text-based threat, vulnerability, and mitigation information: GloVe, BERT and FBERT (a finetuned BERT). We find modest, statistically insignificant, differences. This is likely partly due to the sparsity of linked examples in the large volume of entries in *BRON*. Another reason could be that the linguistic complexity and specific vocabulary of the entries are indistinguishable to the LEMs.
- Our results indicate a sweet spot between entries that only directly frame the relationship, and entries linked too afar from the relationship. This arguably dodges insufficiency on one hand and irrelevance on the other.

Future work should consider how to improve LEM performance, e.g., additional security text sources, such as reports, could be tapped for finetuning. Examining what features contributed to misclassification would yield additional insights. In addition, there are newer and improved LEM transformer models, e.g. RoBERTa and FastText, that might be able to improve the performance. Finally, a process for manually vetting the trustworthiness of the links the classifier predicts is required.

Acknowledgments. This material is based upon work supported by the DARPA Advanced Research Project Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under Contract No. N66001-18-C-4036

REFERENCES

[1] Neda AfzaliSeresht, Yuan Miao, Qing Liu, Assefa Teshome, and Wenjie Ye. 2020. Investigating cyber alerts with graph-based analytics and narrative visualization. In *2020 24th International Conference Information Visualisation (IV)*. IEEE, 521–529.

[2] ALFA Group. 2022. BRON repository. https://github.com/ALFA-group/BRON

[3] Benjamin Ampel, Sagar Samtani, Steven Ullman, and Hsinchun Chen. 2021. Linking Common Vulnerabilities and Exposures to the MITRE ATT&CK Framework: A Self-Distillation Approach. *arXiv e-prints* (2021), arXiv–2108.

[4] Masaki Aota, Hideaki Kanehara, Masaki Kubo, Noboru Murata, Bo Sun, and Takeshi Takahashi. 2020. Automation of Vulnerability Classification from its Description using Machine Learning. *2020 IEEE Symposium on Computers and Communications (ISCC)* (2020), 1–7.

[5] Frederico Araujo, Dhilung Kirat, Xiaokui Shu, Teryl Taylor, and Jiyong Jang. 2021. Evidential Cyber Threat Hunting. arXiv:2104.10319 [cs.CR]

[6] Siddhartha Das, Edoardo Serra, M. Halappanavar, A. Pothen, and E. Al-Shaer. 2021. V2W-BERT: A Framework for Effective Hierarchical Multiclass Classification of Software Vulnerabilities. *ArXiv* abs/2102.11498 (2021).

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[8] Neil Dhir, Henrique Hoeltgebaum, Niall Adams, Mark Briers, Anthony Burke, and Paul Jones. 2021. Prospective Artificial Intelligence Approaches for Active Cyber Defence. arXiv:2104.09981 [cs.CR]

[9] Aviad Elitzur, Rami Puzis, and Polina Zilberman. 2019. Attack Hypothesis Generation. In *2019 European Intelligence and Security Informatics Conference (EISIC)*. IEEE, 40–47.

[10] Ibrahim Ghafir, Mohammad Hammoudeh, Vaclav Prenosil, Liangxiu Han, Robert Hegarty, Khaled Rabie, and Francisco J Aparicio-Navarro. 2018. Detection of advanced persistent threat using machine-learning correlation analysis. *Future Generation Computer Systems* 89 (2018), 349–359.

[11] Yongyan Guo, Zhengyu Liu, Cheng Huang, Jiayong Liu, Wangyuan Jing, Ziwang Wang, and Yanghao Wang. 2021. CyberRel: Joint Entity and Relation Extraction for Cybersecurity Concepts. In *International Conference on Information and Communications Security*. Springer, 447–463.

[12] Casey Hanks, Michael Maiden, Priyanka Ranade, Tim Finin, and Anupam Joshi. [n.d.]. Recognizing and Extracting Cybersecurity Entities from Text.

[13] Erik Hemberg, Jonathan Kelly, Michal Shlapentokh-Rothman, Bryn Reinstadler, Katherine Xu, Nick Rutar, and Una-May O'Reilly. 2020. BRON–Linking Attack Tactics, Techniques, and Patterns with Defensive Weaknesses, Vulnerabilities and Affected Platform Configurations. *arXiv preprint arXiv:2010.00533* (2020).

[14] Erik Hemberg and Una-May O'Reilly. 2021. Using a Collated Cybersecurity Dataset for Machine Learning and Artificial Intelligence. *ArXiv* abs/2108.02618 (2021).

[15] Erik Hemberg, Ashwin Srinivasan, Nick Rutar, and Una-May O'Reilly. 2022. Using Machine Learning to Infer Plausible and Undetected Cyber Threat, Vulnerability and Mitigation Relationships. In *ICML-ML4Cyber ICML Workshop on Machine Learning for Cybersecurity at ICML'22*.

[16] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python. (2020). https://doi.org/10.5281/zenodo.1212303

[17] Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. 2017. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In *Proceedings of the 33rd Annual Computer Security Applications Conference*. 103–115.

[18] Peter E Kaloroumakis and Michael J Smith. [n.d.]. Toward a Knowledge Graph of Cybersecurity Countermeasures. ([n. d.]).

[19] Kenta Kanakogi, Hironori Washizaki, Yoshiaki Fukazawa, Shinpei Ogata, Takao Okubo, Takehisa Kato, Hideyuki Kanuka, Atsuo Hazeyama, and Nobukazu Yoshioka. 2022. Comparative Evaluation of NLP-Based Approaches for Linking CAPEC Attack Patterns from CVE Vulnerability Information. *Applied Sciences* 12, 7 (2022), 3400.

[20] T. H. Le, Huaming Chen, and M. A. Babar. 2021. A Survey on Data-driven Software Vulnerability Assessment and Prioritization. *ArXiv* abs/2107.08364 (2021).

[21] MITRE. [n.d.]. ATT&CK Matrix for Enterprise. https://attack.mitre.org/ https://attack.mitre.org/.

[22] MITRE. [n.d.]. Common Attack Pattern Enumeration and Classification. https://capec.mitre.org/ https://capec.mitre.org/.

[23] MITRE. [n.d.]. Common Vulnerabilities and Exposure. https://cve.mitre.org/ https://cve.mitre.org/.

[24] MITRE. [n.d.]. Common Weakness Enumeration. https://cwe.mitre.org/ https://cwe.mitre.org/.

[25] MITRE. 2021. MITRE Engage. https://engage.mitre.org/

[26] Stephen Frank Moskal. 2021. *HeAt PATRL: Network-Agnostic Cyber Attack Campaign Triage with Pseudo-Active Transfer Learning*. Ph.D. Dissertation. Rochester Institute of Technology.

[27] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[29] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[30] Aditya Pingle, Aritran Piplai, Sudip Mittal, Anupam Joshi, James Holt, and Richard Zak. 2019. Relext: Relation extraction using deep learning approaches for cybersecurity knowledge graph improvement. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 879–886.

[31] Xinda Wang, Shu Wang, Kun Sun, A. Batcheller, and S. Jajodia. 2020. A Machine Learning Approach to Classify Security Patches into Vulnerability Types. *2020 IEEE Conference on Communications and Network Security (CNS)* (2020), 1–9.

[32] Thomas Wolf and et. al. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Empirical Methods in Natural Language Processing: System Demonstrations*.

[33] Peng Zeng, Guanjun Lin, Lei Pan, Yonghang Tai, and J. Zhang. 2020. Software Vulnerability Analysis and Discovery Using Deep Learning Techniques: A Survey. *IEEE Access* 8 (2020), 197158–197172.

A APPENDIX

A.1 Experimental configurations

Data. Figure 3 show a statistics of *BRON*'s entries from 2nd Dec 2021. Only CVEs in 2021 were considered for the weakness vulnerability relationship. There were 174,835 CVE entries in total, and 14,613 entries in 2021.

LEM. The four LEMs we use are:

Bag-of-Words (BoW): A piece of text is represented as a vector containing the count of each token in it along with zero counts for tokens within a background corpus that do not appear in the piece of text. The `CountVectorizer` module was applied to generate the vectors of word counts [28]. This simple baseline ignores meaning and how tokens appear in context.

GloVe [16]: Token embeddings are created with the GloVe model [29]. This so-called word2vec model represents each token, i.e., creates a text embedding, as a numerical vector within an embedding space. To generate the embedding for a piece of text from GloVe, spaCy averages the text embeddings generated by GloVe. A GloVe model is pre-trained to learn the embedding space by being statistically focused upon token co-occurrences in overlapping areas of text. We generate embeddings using the `en_core_web_lg` pipeline in the spaCy library [16].

BERT [7]: BERT (Bidirectional Encoder Representations from Transformers) is our most sophisticated LEM. It has been pre-trained by Google to consider the context of tokens within a piece of text. Pre-training uses (1) "masked language modeling", in which a proportion of tokens are masked and BERT is trained to predict them based on the remaining tokens, and (2) "next sentence prediction", in which BERT is trained to predict whether a given sentence follows another sentence. We obtain GloVe and a tokenizer from the Hugging Face Transformers library [32]. We use both the `BertModel`, so that the pooler output and [CLS] final hidden state could be accessed, and Hugging Face's `BertForMaskedLM`, so that the BERT model could also be finetuned on the masked language modeling objective.

Finetuned BERT (FBERT) [7]: Finetuning BERT on text associated with our problem is potentially an effective method of producing domain-specific embeddings. We finetuned BERT on *BRON* text data, including CWEs, CAPECs, techniques, tactics, mitigations, and detections from the text fields in Table 7, using the masked language modeling objective. A 90/10 train-validation split was applied to the text samples used to finetune. This split was chosen so that the quantity of text available for finetuning was maximized, while also supporting a validation set to evaluate model-fit at each epoch of finetuning. Hugging Face's `DataCollatorForLanguageModeling` [32] was used to batch the training/validation data, pad the sequences to the maximum length of the batch, and randomly mask 15% of the tokens in the sequences. The model was finetuned for 50 epochs with a batch size of 8. After finetuning, we selected the model with the lowest validation loss across all epochs to use in our experiments.

A.2 Classifier configuration

To account for bias in the model algorithm and the data, trials using different seeds were performed for both the classifier and the train-test split. 100 trials were performed for each (embedding, data subset) pair, with seeds in $[0, \ldots, 99]$ used in the `random_state` parameters of the classifier and the data split.

To emphasize the minimization of false positives, we empirically tuned the class error weights of the cost matrices. Using a class weight of 5 on negative examples (and 1 on positive examples) with BoW reduced the false positive rate, while also increasing accuracy. In fact, when inferring edges, using the class weight reduced the proportion of results with probability above 0.5 from 29% to 19%. However, when we increased the class weight of negative examples and used the `RandomForestClassifier` on input embeddings from GloVe or BERT, the false positive rate increased. Ultimately, we used, for edge inference, with a class weight of 5 on negative examples when BOW embeddings were inputs, and a default class weight of 1 when GloVe and BERT embeddings were used to train `RandomForestClassifier`.

A.3 Result Analysis

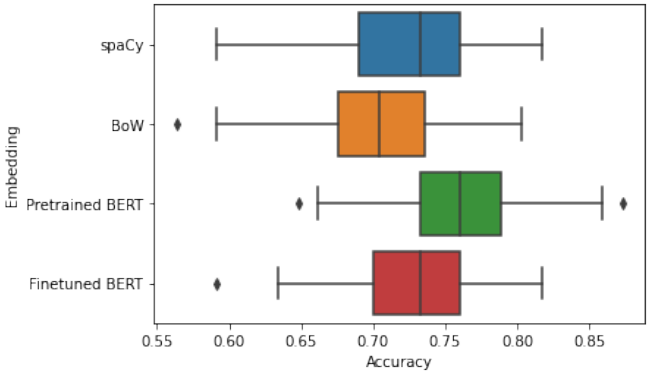
CWE-CAPEC (Figure 4e). GloVe(labeled spaCy) has highest median for D3FEND-Technique, CWE-CVE (Figure 4d) and Engage-Technique (Figure 4c). Note that Engage-Technique and CWE-CVE median values for the top LEMs are quite close.

LEM Embedding analysis. Embeddings of the text associated with the relationship candidates were reduced to 2 dimensions using principal component analysis (PCA) so that they can be visualized. Overall, it seems that there are several pairs of tokens that are similar by observation as well as by their embeddings.

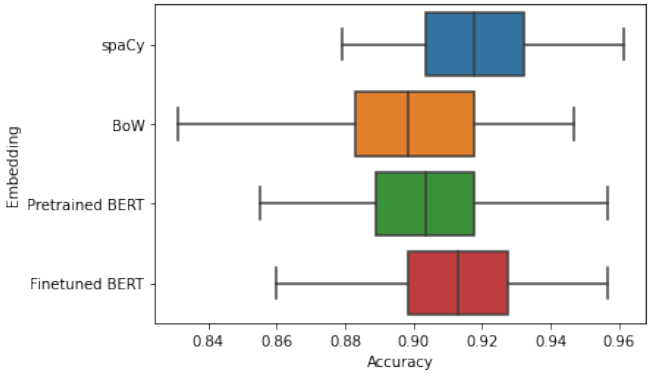
Figure 5 shows PCA plots of the text embeddings in the CAPEC-Technique relationship candidates using GloVe and pre-trained BERT. In both embeddings, it seems that similar texts are relatively close to each other, such as "Password Guessing" and "Password Brute Forcing" in GloVe and "Services Footprinting" and "Account Footprinting" in BERT. Also, there are some fairly similar CAPEC and Technique embeddings that are also relationship candidates, such as "Create files with the same name as files protected with a higher classification" and "Deobfuscate/Decode Files or Information", and "Kerberoasting" and "NTDS".

Table 7: Examples of text in BRON entry types from the fields used in our experiments.

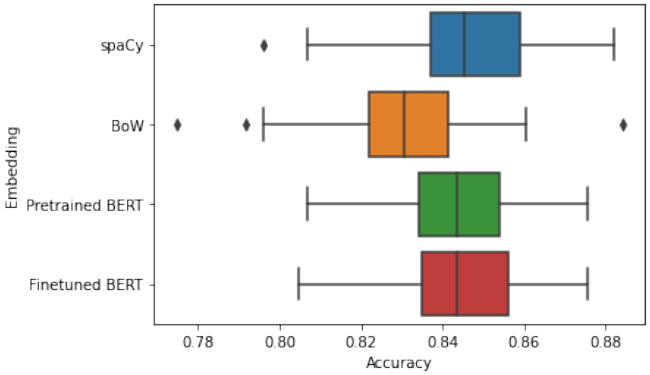
Type	Example
Tactic (τ)	collection
Technique (T)	Data Obfuscation
CAPEC (A)	Accessing Functionality Not Properly Constrained by ACLs
CWE (W)	Sensitive Cookie Without 'HttpOnly' Flag
CVE (V)	ip_input.c in BSD-derived TCP/IP implementations allows remote attackers to cause a denial of service (crash or hang) via crafted packets.
Engage Activity (E_A)	API Monitoring
D3FEND (M_D)	Active Certificate Analysis
Technique mitigation (M_T)	Application Developer Guidance
CAPEC mitigation (M_A)	Implement intelligent password throttling mechanisms such as those which take IP address into account, in addition to the login name.
CWE mitigation (M_W)	The application configuration should ensure that SSL or an encryption mechanism of equivalent strength and vetted reputation is used for all access-controlled pages.
Technique detection (D_T)	Data Obfuscation
CAPEC detection (D_A)	The only indicators of successful Blind SQL Injection are the application or database logs that show similar queries with slightly differing logical conditions that increase in complexity over time. However, this requires extensive logging as well as knowledge of the queries that can be used to perform such injection and return meaningful information from the database.
CWE detection (D_W)	This specific weakness is impossible to detect using black box methods. While an analyst could examine memory to see that it has not been scrubbed, an analysis of the executable would not be successful. This is because the compiler has already removed the relevant code. Only the source code shows whether the programmer intended to clear the memory or not, so this weakness is indistinguishable from others.



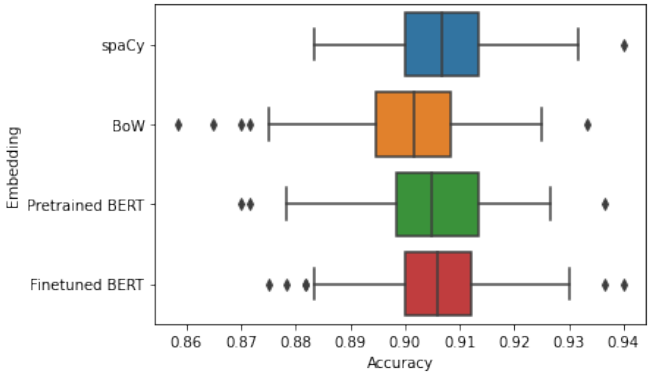
(a) CAPEC-Technique



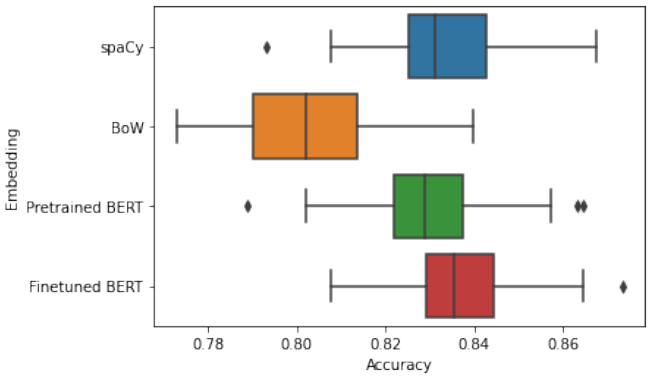
(b) D3FEND-Technique



(c) Engage-Technique

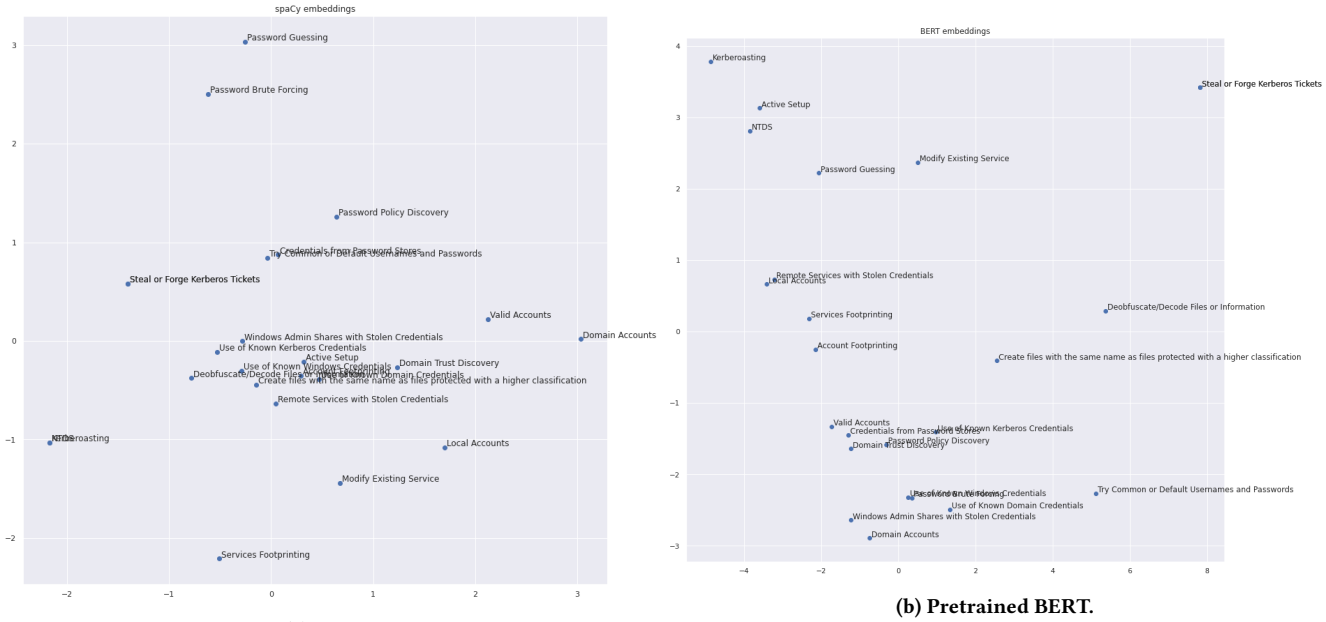


(d) CWE-CVE



(e) CWE-CAPEC

Figure 4: Performance comparison of each LEM for each relationship. Results are for 100 repetitions with the best dataset, see Table 6. Note that the x-axis scales are different for readability of differences between methods for each relationship. Note spaCy refers to GloVe



(a) GloVe. **(b) Pretrained BERT.**

Figure 5: PCA plots of text embeddings of the CAPECs and Techniques in the CAPEC-Technique edge candidates.