Norwegian University
of Life Sciences

Master's Thesis 2024   30 ECTS
Faculty of Science and Technology

Knowledge Graphs for Software Security Assessments and Cyber Threat Intelligence.

Sougata Bhattacharya
MSc Data Science

Knowledge Graphs for Software Security Assessments and Cyber Threat Intelligence.

Sougata Bhattacharya

Faculty of Science and Technology
Norwegian University of Life Sciences

May 14, 2024

Acknowledgement

This thesis has only been possible with the support, guidance, and encouragement of several individuals and institutions. It is with deep gratitude that I acknowledge their role in this endeavour.

I would like to start by thanking my employer, Storebrand, for agreeing to my audacious proposition of studying for a Master's degree along with full-time work.

My supervisor Prof. Eirik Valseth has consistently gone above and beyond to guide me, expediting processes that would have otherwise been time-consuming. We had very fruitful discussions, and his reassuring demeanour always had a positive effect on me.

Special mention to my co-supervisor Anders Mølmen Høst, whose patience and wisdom guided me through my uncertainties. His ability to simplify complex concepts and provide clarity in trying times was invaluable to my research.

I would also like to mention my other co-supervisor Prof. Leon Moonen, whose continued guidance and support, even in the face of personal adversity, has been truly inspiring.

I am grateful to SIMULA and NMBU for providing access to their resources, including the eX^3 and Orion supercomputers.

Finally, I express my profound gratitude to my wife Payel, for her unfailing support, continuous encouragement and guidance throughout the years of study and through researching and writing this thesis. This accomplishment would not have been possible without her.

Abstract

In the dynamic field of cybersecurity, the identification and mitigation of software vulnerabilities are critical for safeguarding digital infrastructures. This thesis explores the integration of *Knowledge Graphs* with language modeling techniques to enhance security assessments of *Cyber Threat Intelligence* reports. The research delves into the challenges faced by traditional machine learning approaches in predicting attack techniques from these reports and proposes a methodology that combines the strengths of *Pretrained Language Models* and *Knowledge Embeddings* to improve predictive accuracy. The study, utilizing the *Threat Report ATT&CK Mapper* dataset, expands on the *Knowledge Embedding and Pre-trained LanguagE Representation* model to encode textual Cyber Threat Intelligence descriptions into *Knowledge Graph* triples for training the *Knowledge Embedding* objective, while simultaneously using the same descriptions for training a *Masked Language Model* objective. The two top-performing models from this study show better Precision, Recall, and F1 scores than the Threat Report ATT&CK Mapper tool when trained and evaluated on the same dataset. These findings suggest that the proposed approach is a viable method of predicting Attack Techniques from CTI reports. The thesis presents a practical approach to the application of Knowledge Graphs for cybersecurity, offering a framework for the automated analysis of cyber threats.

Contents

Acronyms

ABox Assertional Box. 11, 14

BERT Bidirectional Encoder Representations from Transformers. 8, 10, 20, 21, 24

BPE Byte-Pair Encoding. 10, 31

CISA Cybersecurity & Infrastructure Security Agency. 2

CKG Cybersecurity Knowledge Graphs. 17–19, 72

CTI Cyber Threat Intelligence. 2–6, 12, 13, 17–19, 21, 31, 34, 37, 38, 57, 59, 61, 62, 68, 70, 71

CVSS Common Vulnerability Scoring System. 6

CWE Common Weakness Enumeration. 6

ENISA European Union Agency for Cybersecurity. 2

F1 F1 Score. 15, 16, 60, 66, 67, 70, 71

FAIR Fundamental AI Research. 19

FAIRSEQ Facebook AI Research Sequence-to-Sequence. 19, 20, 22, 28, 32

FN False negative. 15, 16

FP False positive. 15, 16

GCN Graph Convolutional Neural Networks. 18, 72

GLUE General Language Understanding Evaluation. 21

GPT Generative Pre-trained Transformer. 8, 9, 31

ISACs Information sharing and analysis centers. 2

KB Knowledge Base. 4, 11, 27

KE Knowledge Embedding. 4, 5, 10, 11, 22, 23, 25, 31, 32, 37, 38, 41, 45, 47, 51, 69–71, 73, 74

KEPLER Knowledge Embedding and Pre-trained LanguagE Representation. 5, 19, 22–25, 28, 31, 32, 34, 37, 38, 41–43, 55–57, 59, 61–63, 66, 67, 70–72

KG Knowledge Graph. 3–6, 10–12, 14, 18, 19, 22, 23, 25, 32, 37, 70, 72, 73

LLMs Large Language Models. 9, 17, 19, 21, 72, 73

LSI Latent Semantic Indexing. 18

MITRE ATT&CK MITRE Adversarial Tactics, Techniques, and Common Knowledge. 1, 3, 6, 14, 18, 19, 27, 29, 39, 40, 70

MLM Masked Language Model. 9, 10, 22, 23, 31, 32, 37, 38, 41, 45, 47, 51, 69

MR Mean rank. 25

MRR Mean reciprocal rank. 25

NIST National Institute of Standards and Technology. 2

NLG Natural Language Generation. 7, 8

NLP Natural Language Processing. 7–10, 17, 20–23, 32

NLU Natural Language Understanding. 7

NSP Next Sentence Prediction. 20

NVD National Vulnerability Database. 1, 6

OWASP Open Web Application Security Project. 6

P Precision. 15, 16, 60, 66, 67, 70–72

PLMs Pretrained Language Models. 4, 8, 9, 22, 74

PyG Pytorch Geometric. 32

R Recall. 15, 16, 60, 65–67, 70–72

RACE Reading Comprehension Dataset From Examinations. 21

RoBERTa A Robustly Optimized BERT Pretraining Approach. 8, 9, 19–22, 24, 25

SQuAD Stanford Question Answering Dataset. 21

STIX 2.1 Structured Threat Information Expression. 3

TBox Terminological Box. 11–13

TFIDF Term Frequency-Inverse Document Frequency. 18

TN True negative. 15

TP True positive. 15, 16

TRAM Threat Report ATT&CK Mapper. 4, 5, 14, 19, 21, 27, 29, 32–34, 36–38, 40, 42, 53, 55, 56, 58, 62–64, 66–68, 70–72

TTPs tactics, techniques, and procedures. 2–4, 6, 21, 27

Chapter 1

Introduction

In the ever-evolving landscape of cybersecurity, identifying and mitigating software vulnerabilities is of paramount importance [1]. Software vulnerabilities represent weaknesses within software systems that malicious actors can exploit to trigger unintended and potentially harmful actions [2].

Addressing software vulnerabilities is an ongoing challenge. They range from critical flaws in widely used operating systems to subtle issues in specialized applications. Exploiting these vulnerabilities can lead to a cascade of consequences, affecting not only the security and privacy of individuals and organizations but also having widespread economic and societal implications, with scenarios ranging from data breaches and identity theft to the compromise of critical infrastructure and massive financial losses. To mitigate these risks effectively, it is essential to discover, assess, and remediate software vulnerabilities promptly and comprehensively.

In recent years, there has been a notable trend in software security assessment towards embracing data-driven methodologies [3]. These methodologies depend on up-to-date repositories that catalog known software vulnerabilities, utilizing this information to streamline the process of detecting vulnerabilities in current software systems. A significant feature of these methodologies is their capability to gather, categorize, and distill knowledge regarding specific vulnerabilities. They draw upon a variety of data sources, including the National Vulnerability Database (NVD)[1], MITRE Adversarial Tactics,

[1]https://nvd.nist.gov/

Techniques, and Common Knowledge (MITRE ATT&CK)[2], along with several Cyber Threat Intelligence (CTI) sources.

As digital infrastructures become increasingly complex and integral to organizational operations, the threats to software security grow both in sophistication and frequency. CTIs has emerged as a crucial element in the cybersecurity arsenal. It is a process of collection and analysis of information about current and potential attacks that threaten the safety of an organization's or individual's assets [4]. It is through CTI reports that organizations gain actionable insights into the tactics, techniques, and procedures (TTPs) of threat actors, enabling them to anticipate and mitigate cyber threats effectively. *Tactics* are the high-level descriptions of the behavior and strategy of a threat actor, including the overall objectives and goals of their attacks. *Techniques* stand for the general methods or types of activity that attackers use to carry out their tactics, and can involve several steps. *Procedures* are the specific, detailed steps that threat actors use to execute a technique, often including exact tools or methods used in an attack.

CTI reports are typically generated by cybersecurity teams within organizations, specialized CTI providers, or government agencies. Some reports are produced in response to specific incidents or emerging threats, while others are generated on a regular schedule. Entities that produce CTI reports include private cybersecurity firms, industry-specific Information sharing and analysis centers (ISACs) like European Union Agency for Cybersecurity (ENISA)[3], government bodies like the National Institute of Standards and Technology (NIST)[4], etc. One such list of current CTI reports could be found in Cybersecurity & Infrastructure Security Agency (CISA)[5].

Understanding and analysis of CTI reports are of paramount importance; they are the linchpin that connects the dots between disparate pieces of cybersecurity data. These reports not only provide a retrospective view of cyber incidents but also offer a prospective outlook on potential vulnerabilities and emerging threats. However, the challenge lies in the complexity and volume of data that CTI encompasses [4].

[2]https://attack.mitre.org/
[3]https://www.enisa.europa.eu/topics/national-cyber-security-strategies/information-sharing
[4]https://www.nist.gov/
[5]https://www.cisa.gov/news-events/cybersecurity-advisories

While these data sources are invaluable for identifying and assessing vulnerabilities, they often maintain their data in ways that are not inherently machine-actionable or interoperable [5]. An example of this is the MITRE ATT&CK dataset[6], which is represented in the Structured Threat Information Expression (STIX 2.1)[7] JSON collections. While STIX 2.1 provides a machine-readable format for accessing the MITRE ATT&CK knowledge base, there is no straightforward method to convert this data into PyTorch Geometric [6], a state-of-the-art graph structure. This lack of semantic interoperability and structure hampers the ability to harness the full potential of this wealth of data automatically.

This is where Knowledge Graph (KG) (Section 2.1.4) have emerged as a solution to bridge the gap between disparate data sources and facilitate the automation of software security assessments and Cyber Threat Intelligence [7]. They offer a structured, dynamic way of representing and analyzing the vast array of interconnected data points found within CTI reports [8, 9]. A KG is represented as a collection of interlinked entities, e.g., objects, events, or concepts. These entities are depicted as `nodes`, and the relationships between them are depicted as the `edges`, forming a network of interconnected information. A fundamental unit of this structure is the `triple`, which consists of a `subject`, a `predicate`, and an `object`. The `subject` represents the entity, the `predicate` describes the relationship, and the `object` is the entity to which the subject is related.

A Cybersecurity Knowledge Graph serves as a foundational asset for collecting, organizing, and reasoning about cyber-related information [10]. Such a Knowledge Graph combines knowledge about known software vulnerabilities with information about common attack TTPs employed by cybercriminals. By creating a cybersecurity Knowledge Graph, we aim to facilitate automated reasoning, decision-making, and understanding of the relationships within the realm of software security.

[6]https://github.com/mitre-attack/attack-stix-data

[7]https://docs.oasis-open.org/cti/stix/v2.1/csprd01/stix-v2.1-csprd01.html

1.1 Research questions & Hypothesis

Machine learning models require large amounts of high-quality, labeled data to train effectively. In the context of cyber threat intelligence, such data can be scarce, reliant on extensive manual effort, and may not be labeled consistently, leading to challenges in training accurate models.

Which leads to our **research questions**:

- What challenges or limitations do traditional machine learning approaches encounter when attempting to predict attack tactics, techniques, and procedures from CTI reports?
- To what extent would the incorporation of Knowledge Graphs alongside current machine learning methods improve a model's predictive accuracy for attack TTPs?

It is known that the consistency of Pretrained Language Models (PLMs) is generally poor with respect to factual knowledge [11]. This has led to open questions about PLMs on their potential to function as Knowledge Base (KB).

On the other hand, Knowledge Embedding (KE) methods have demonstrated high effectiveness in mapping relational facts within Knowledge Graph (KG)s through the utilization of entity embeddings, although they do not match the performance of PLMs in directly extracting information from text [12, 13, 14].

The basis of our input data for our work is taken from Threat Report ATT&CK Mapper (TRAM)[8](Section 2.3.3), which consists of CTI document titles and texts, labeled with attack techniques. The most recent TRAM2 annotation effort[9], focused on identification of TTPs from CTI reports based solely on PLMs. We build on the work of Wang et al. [14] with the **hypothesis** that training a PLM in conjunction with a KE model on the pre-labeled CTI data could leverage the advantages of both models.

To address our research questions and evaluate our hypothesis, we adopt

[8]https://github.com/center-for-threat-informed-defense/tram/tree/main/data/tram2-data

[9]https://medium.com/mitre-engenuity/our-tram-large-language-model-automates-ttp-identification-in-cti-reports-5bc0a30d4567

the following approach, which is detailed in the next few chapters. It involves encoding textual CTI descriptions from TRAM training data using a PLM as their embedding, transforming them into KG triples, and simultaneously optimizing Knowledge Embedding and Language Modelling objectives by extending on the work done in Knowledge Embedding and Pre-trained LanguagE Representation (KEPLER)[14](Section 2.3.4), to assess the performance of this method.

Chapter 2

Theory and Related work

The current cybersecurity landscape is marked by an increasing complexity of threats and the need for more sophisticated tools to detect and mitigate them. Knowledge Graphs can address this need by combining information about known software vulnerabilities from CTI reports with insights into attack TTPs. [15, 16]

In the past few years, the cybersecurity community has increasingly focused on the development and use of security Knowledge Graphs from various data sources such as the NVD[10], MITRE ATT&CK[11], CTIs, Common Weakness Enumeration (CWE)[12], Common Vulnerability Scoring System (CVSS)[13], Open Web Application Security Project (OWASP)[14], etc. These graphs are then enriched with additional knowledge extracted using machine learning, natural language processing, and other data-driven techniques. The goal is to automate the identification and classification of actively exploited vulnerabilities, thereby enhancing the capability to mitigate threats.

[10]https://nvd.nist.gov/
[11]https://attack.mitre.org/
[12]https://cwe.mitre.org/
[13]https://www.first.org/cvss/
[14]https://owasp.org/

2.1 Theory

In this section, we discuss some of the most relevant concepts that form the foundational building blocks of our work.

2.1.1 Natural Language Processing

Natural Language Processing (NLP) as a field that lies at the confluence of computer science, artificial intelligence, and linguistics [17]. It aims to bridge the gap between human communication and computer understanding. Within NLP, two primary subfields emerge: Natural Language Understanding (NLU) and Natural Language Generation (NLG). These subfields represent the dual nature of language processing, encompassing both the interpretation and the production of language.

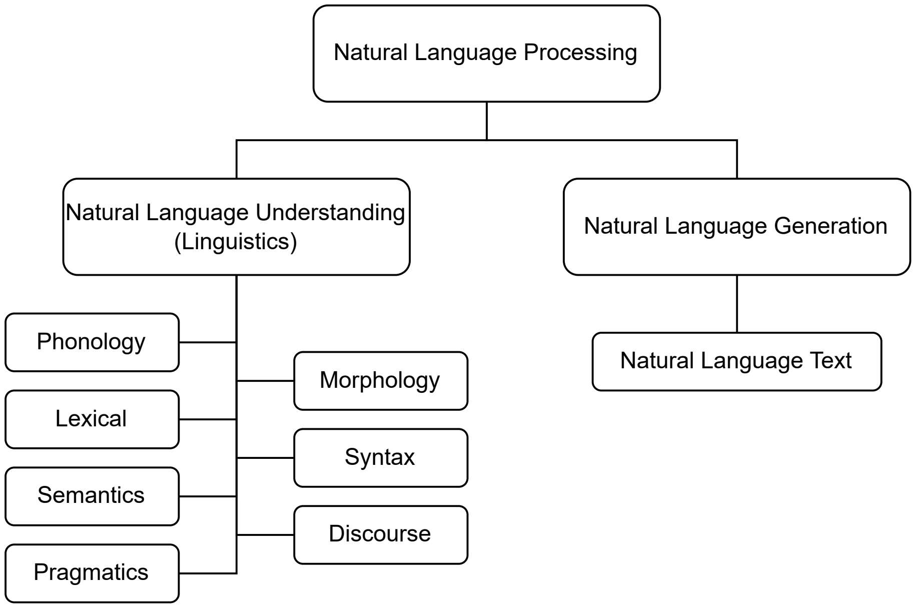
Natural Language Processing
Natural Language Understanding (Linguistics)
Natural Language Generation
Phonology
Lexical
Semantics
Pragmatics
Morphology
Syntax
Discourse
Natural Language Text

Figure 2.1: Broad classification of NLP [17].

Natural Language Understanding is the subfield dedicated to the comprehension of language. It involves a variety of linguistic components, each playing a crucial role in how meaning is derived from text or speech. `Phonology`, the study of sound patterns, is foundational to recognizing spoken words and their nuances. `Morphology` goes a step further by analyzing the structure of words, identifying how roots combine with prefixes and suffixes to form new meanings. `Lexical` analysis involves the understanding of the meaning of each word. `Syntax` examines the rules that govern word order, ensuring that sentences are not only well-formed but also meaningful. `Semantics` then takes these sentences and extracts meaning, considering the various possible interpretations of words within different contexts. `Discourse` analysis goes beyond individual sentences to examine larger linguistic structures, ensuring textual coherence by interpreting relationships between sentences and resolving references within the text. Lastly, `pragmatics` looks beyond the literal meanings to understand the implied intentions and social cues embedded in language use. Collectively, these elements allow machines to interpret and comprehend human language in a manner that is both significant and pertinent to the context.

Natural Language Generation, on the other hand, focuses on the production of language. It is the process by which structured data is transformed into natural language. This transformation involves several steps, beginning with the planning of content, where the system decides what information to include and how to organize it logically. The next step is sentence realization, where the system converts the structured data into fluent natural language sentences. This process is not merely about stringing words together; it requires a deep understanding of grammar, style, and coherence to produce text that reads naturally.

NLP has come a long way since its inception in the 1940s. Early efforts focused on rule-based systems, but recent advancements leverage machine learning and deep learning techniques. Notable milestones include:

1950s The birth of computational linguistics and early rule-based translation systems.

1980s The emergence of statistical methods and the development of parsers.

2000s The rise of machine learning models [18].

2010s Deep learning revolution with neural networks, attention mechanisms, and Pretrained Language Models (e.g., BERT [19]).

Recent breakthroughs include models like BERT [19], GPT [20] and RoBERTa [21] (Section 2.3.2), which pre-trains on vast amounts of unlabeled text and finetunes for specific NLP tasks. Their contextual embeddings capture rich language semantics and have significantly improved performance across various applications. However, evaluating NLP systems remains challenging due to context, ambiguity, and domain-specific nuances.

2.1.1.1 Pretrained Language Models & Large Language Models

Pretrained Language Models (PLMs) have been trained on large text corpora to learn a wide array of language features and patterns before being fine-tuned for specific tasks. Large Language Models (LLMs), such as GPT-3 [20] and RoBERTa [21], are subsets of these models characterized by their vast number of parameters, allowing them to understand and generate human-like text.

Language models were traditionally task-specific and trained from scratch. The shift towards pre-trained models, trained on extensive text datasets, allows for general language representations that can be fine-tuned for specific tasks.

The **core concept** of these models is the idea of distributed representations, where words and phrases are encoded as vectors in a continuous space. This enables the models to capture semantic and syntactic properties through context-dependent representations. Pre-training uses self-supervised learning tasks like Masked Language Models to understand language context while fine-tuning adapts the model to specific NLP tasks.

LLMs scale up PLMs, featuring billions of parameters. They are trained on diverse text datasets, enabling them to generate coherent and contextually rich text, demonstrating remarkable performance on various NLP tasks without task-specific training data [22].

2.1.1.2 Masked Language Models

In the field of NLP, Masked Language Model (MLM)s have emerged as a major innovation in self-supervised learning as an effective pre-training approach. At its core, an MLM is a neural-network based language model specifically trained to predict missing or "masked" words within a given text. The fundamental idea involves temporarily replacing certain words in a sentence with a special token (often denoted as [`MASK`]). The model's task is then to predict the correct words that should replace these masked tokens, leveraging the context provided by the surrounding words [23].

Bidirectional Encoder Representations from Transformers (BERT), a MLM developed by Google AI[15], has revolutionized the understanding of language structure by employing attention mechanisms to grasp the contextual relationships between words. BERT's bidirectional text scanning capability allows it to comprehend the context from both directions, offering substantial improvements in tasks such as sentence completion and language translation [19].

2.1.1.3 Byte Pair Encoding

Byte-Pair Encoding (BPE), introduced by Sennrich, Haddow, and Birch [24], is a technique used in NLP to break down words into smaller, more manageable units, known as subwords.

The BPE algorithm starts by initializing a vocabulary with all unique characters or bytes in a given text. It then calculates the frequency of each character or byte in the text. The algorithm repeatedly identifies the most common pair of consecutive characters or bytes and merges them into a single subword unit. This process continues until the vocabulary reaches a predefined size.

The resulting subwords can efficiently represent the original text, making BPE a valuable tool for tasks such as machine translation, text classification, and text generation. Despite its simplicity, BPE is powerful and reliable, with a low computational cost.

2.1.2 Knowledge Embedding

Knowledge Embedding (KE) is a technique for semantic representation within Knowledge Graph (KG)[25](Section 2.1.4). It involves the translation of complex, high-dimensional information about entities and their relationships into a lower-dimensional, continuous vector space known as the *embedded space*. This transformation is crucial for capturing the essence of *Semantic proximity*, which refers to the closeness of meaning that entities share in the KG, ensuring that related concepts remain close in the embedded space. By doing

[15] https://ai.google/

so, KE enables efficient retrieval, reasoning, and inference over large sets of interconnected data.

An *embedded space* is a mathematical construct where each entity and relationship from the knowledge graph is represented as a point or vector. The goal of Knowledge Embedding is to arrange both the entity and relationship vectors such that the distances between them reflect the semantic proximity of the corresponding entities in the Knowledge Graph.

Semantic proximity implies that entities with similar meanings or roles within the graph are located near each other in the embedded space, facilitating the discovery of semantic relationships and patterns that are not immediately apparent in the high-dimensional original data.

2.1.3 Knowledge Base

A *Knowledge Base* is a structured repository used for knowledge sharing and management. It organizes information into two main components: Terminological Box (TBox) and Assertional Box (ABox) [26].

Terminological Box (TBox): This component defines the terminologies, i.e., the vocabulary of the domain, and the relationships between them. It includes the classes (or concepts), properties (or roles), and the constraints or rules that apply to them. In essence, the TBox represents the schema or the ontology of the knowledge base.

Assertional Box (ABox): This component contains assertions about instances of the classes and properties defined in the TBox. In other words, it holds the actual data or facts that populate the schema defined by the TBox.

2.1.4 Knowledge Graph

A *Knowledge Graph (KG)* [7, 25] is a powerful concept for representing complex systems of entities and their interrelationships. It is essentially a form of *semantic network*, structured as a graph, where nodes represent entities and edges represent relationships or connections between these entities. It enables more efficient data integration, interpretation, and analysis by providing a structured and interconnected view of data.

A *semantic network* is a type of knowledge base that depicts the semantic connections amongst concepts within a network [27]. It is frequently employed as a method for knowledge representation. In a semantic network, elements, which stand for concepts, are depicted as vertices, and edges signify the semantic links between these concepts.

One such example of a KG is WikiData[16], which is the largest general-interest knowledge base that is openly available. It is collaboratively edited by thousands of volunteer editors and has evolved considerably since its inception in 2012 [28].

As noted in previous sections, a Knowledge Graph is represented as a collection of interlinked nodes, and the relationships between them are depicted as the edges, forming a network of *"triples"*. An example triple would look like `("T1548.002", "used in", "Enigma Stealer Targets Cryptocurrency Industry with Fake Jobs")` where "`T1548.002`" is the subject, representing an attack technique; "`used in`" is the predicate, indicating the relation of the subject to the object; and "`Enigma Stealer Targets Cryptocurrency Industry with Fake Jobs`", the CTI report name, is the object.

To materialize this KG, the following elements could be defined:

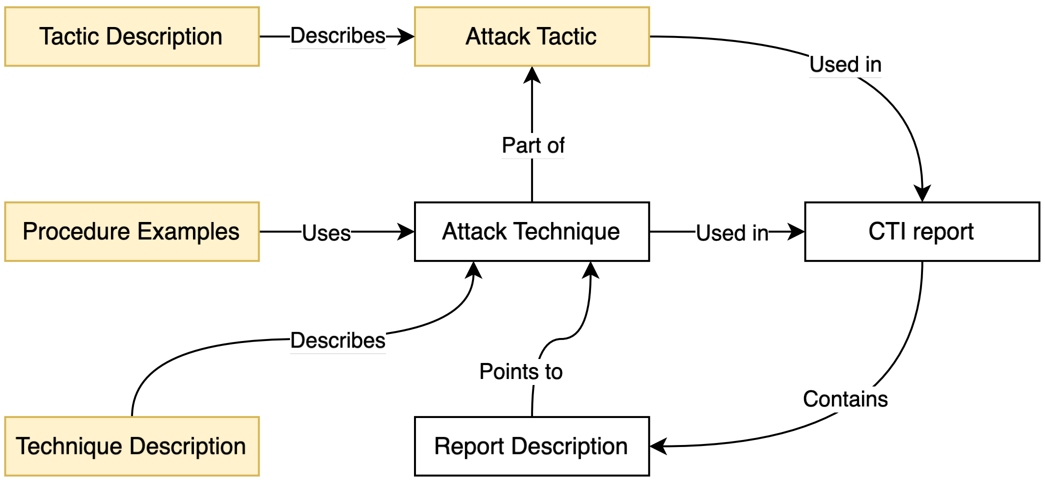

Figure 2.2: TBox : A Knowledge Graph ontology.

Nodes (Entities) of the TBox(Figure 2.2) are as follows:

[16]https://www.wikidata.org/

- *Attack Tactic*[17]: Signifies "why" or the motive behind an adversary's actions.
- *Tactic Description*: Contains the full description of the tactic.
- *Attack Technique*[18]: Depicts the methods used by adversaries to accomplish their strategic objectives through the execution of specific tasks.
- *Technique Description*: Contains the full description of the technique.
- *CTI report*: The name of a Cyber Threat Intelligence report.
- *Report Description*: Text excerpts from a CTI report.
- *Procedure examples*: The particular application or real-world utilization of techniques or sub-techniques employed by the adversary.[19]

Edges (Relationships) of the TBox(Figure 2.2) are as follows:

- *Part of*: Links Attack Techniques to Attack Tactics.
- *Used in*: Shows that the techniques and tactics are made use of in a CTI report.
- *Points to*: Links a text to an Attack Technique.
- *Contains*: Connects a text to a CTI report.
- *Uses*: Associates procedure-examples to Attack Techniques.
- *Describes*: Full names of the attack techniques and tactics.

[17] https://attack.mitre.org/tactics/enterprise/
[18] https://attack.mitre.org/techniques/enterprise/
[19] https://attack.mitre.org/resources/faq/

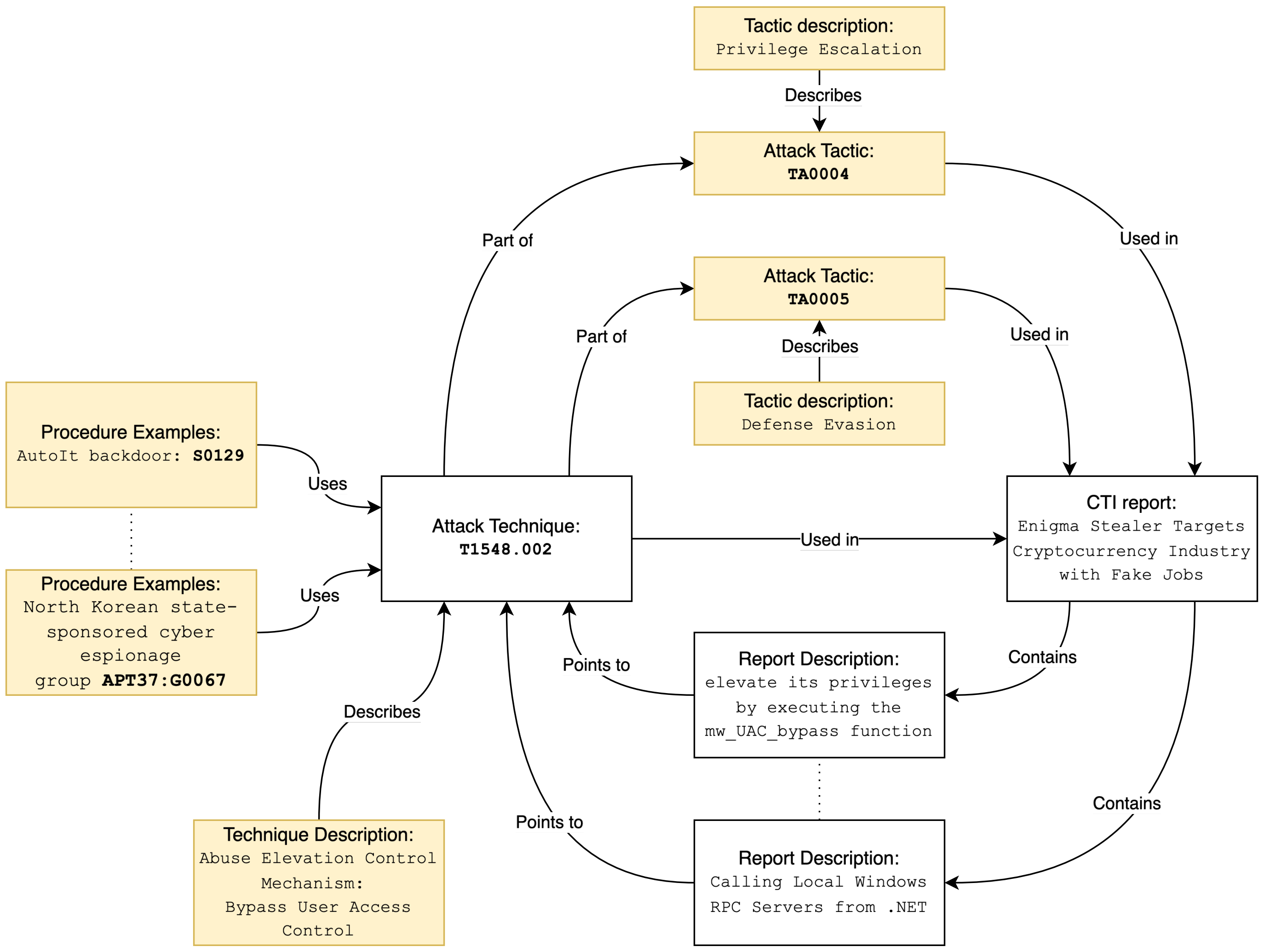
Tactic description: Privilege Escalation
Describes
Attack Tactic: TA0004
Part of
Used in
Attack Tactic: TA0005
Part of
Describes
Used in
Tactic description: Defense Evasion
Procedure Examples: AutoIt backdoor: S0129
Uses
Attack Technique: T1548.002
Used in
CTI report: Enigma Stealer Targets Cryptocurrency Industry with Fake Jobs
Procedure Examples: North Korean state-sponsored cyber espionage group APT37:G0067
Uses
Points to
Report Description: elevate its privileges by executing the mw_UAC_bypass function
Contains
Describes
Technique Description: Abuse Elevation Control Mechanism: Bypass User Access Control
Points to
Report Description: Calling Local Windows RPC Servers from .NET
Contains

Figure 2.3: ABox: A Knowledge Graph assertion.

The ABox(Figure 2.3) shows an implementation of the ontology(Figure 2.2). *Note: The yellow-highlighted entities in Figure 2.2 and Figure 2.3 represent additional information that was extracted from MITRE ATT&CK data and incorporated into the TRAM data to enhance its relational information, thereby potentially improving the model performance. Detailed description of this in Section 3.2.2*

2.1.5 Prediction Scoring Metrics

A **Confusion Matrix** is a specific table layout that allows visualization of the performance of a supervised learning model. It's extremely useful for measuring metrics like Precision (P), Recall (R), and F1 Score (F1). It is a table with two sets of rows & columns that show the False positive (FP), False negative (FN), True positive (TP), and True negative (TN) counts. This allows more detailed analysis than a mere proportion of correct classifications (accuracy).

- TP are the correctly predicted positive values.
- TN are the correctly predicted negative values.
- FP occur e.g., when an actual class is *no* and the predicted class is *yes.*
- FN are the cases when, e.g., the actual class is *yes* but the predicted class in *no.*

A Confusion Matrix can be visualized as follows:

	Predicted: Yes	Predicted: No
Actual: Yes	TP	FN
Actual: No	FP	TN

Table 2.1: A Confusion Matrix

Precision refers to the ratio of correctly predicted positive instances to the total number of predicted positive instances. It shows the model's capability to accurately classify positive instances. It's calculated as:

$$P = \frac{TP}{TP + FP}$$

Recall, also known as sensitivity, is the ratio of correctly identified positive instances to all actual positive instances. It demonstrates the model's ability to correctly detect positive instances. It is defined as:

$$Recall = \frac{TP}{TP + FN}$$

The **F1 Score** is the harmonic mean of P and R where precision and recall are equally weighted, providing a single metric that balances both aspects. It is defined as:

$$F1 = 2 \times \frac{P \times R}{P + R}$$

Micro-Averaging: This approach gives equal importance to each instance or data point. The metric is computed on a global scale by tallying the total number of TPs, FNs, and FPs. This technique is beneficial when the goal is to assign equal weight to each prediction. This metric is defined by:

$$MicroPrecision = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i + FP_i)}$$

$$MicroRecall = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i + FN_i)}$$

Macro-Averaging: This method calculates the metric independently for each class and then takes the average, thereby ensuring all classes are treated equally. This technique is advantageous when the aim is to evaluate the overall performance of a model, with each class being given equal consideration. This is defined by:

$$MacroPrecision = \frac{1}{n}\sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i}$$

$$MacroRecall = \frac{1}{n}\sum_{i=1}^{n} \frac{TP_i}{TP_i + FN_i}$$

In both Micro-Averaging and Macro-Averaging equations,

- TP_i is the number of True positive for the i^{th} class.
- FP_i is the number of False positive for the i^{th} class.
- FN_i is the number of False negative for the i^{th} class.
- n is the number of classes.

Perplexity is a metric used in language modeling to assess the effectiveness of a model in predicting a sample. Specifically, in NLP, it serves as a measure of the model's ability to predict a text sequence. A language model with a lower perplexity score is considered to perform better in generalization, meaning it has a higher certainty in its predictions.

2.2 Related Work

In this section, we explore a selection of studies relevant to our thesis, which highlight the current state of research.

Building on LLMs, Jin et al. [29]'s survey systematically reviews the use of LLMs in processing graph-structured data. The paper categorizes potential scenarios into pure graphs, text-attributed graphs, and text-paired graphs. It discusses techniques for utilizing LLMs as predictors, encoders, and aligners on graphs, comparing the advantages and disadvantages of different types of models. The survey also addresses real-world applications, providing open-source codes and benchmark datasets, and concludes with potential future research directions in this rapidly evolving field. It discusses the fact that using LLMs as encoders is possibly the most direct approach to leverage LLMs on graphs, and this is a method we employ in our work.

Motlagh et al. [30]'s paper provides a comprehensive review of the applications of LLMs in cybersecurity. It categorizes the literature into defensive[20] and adversarial[21] uses of LLMs, highlighting both the potential risks and opportunities. The study emphasizes the slow adoption of machine learning in cybersecurity and suggests that LLMs can significantly elevate the capabilities in this domain by offering advanced solutions for threat detection and prevention. In our work, we apply LLMs as a defensive measure in identifying Attack Techniques from CTI reports.

Sikos [10] provides a survey on Cybersecurity Knowledge Graphs (CKG), which are graph-based data models representing cyber-knowledge. These CKGs offer holistic approaches for processing vast volumes of complex cybersecurity data from diverse sources. The author discusses the leading graph-based models applied in the field of cybersecurity. Additionally, the author

[20]https://www.nist.gov/cyberframework
[21]https://attack.mitre.org/matrices/enterprise/

delves into systems for organizing knowledge that establish the principles and attributes employed in the structured representation of cyber-related knowledge. This includes both background knowledge and specific expert knowledge about actual systems or attacks. The paper also looks into how CKGs support machine learning and help in automated decision-making on cyber-knowledge, which is crucial for detecting anomalies, categorizing vulnerabilities, and matching attack patterns. The author's work emphasizes the importance of CKGs in achieving an advanced level of cyber-situational awareness, uncovering fresh cyber-knowledge, visualizing data flow, attack routes and networks, and comprehending data correlations through the consolidation and merging of data.

Dasgupta et al. [31]'s work introduces a method to enhance CKGs using Graph Convolutional Neural Networks (GCN). The paper focuses on the challenge of verifying the accuracy of CKG assertions, which are represented as semantic triples. To address this, the authors propose a supervised machine-learning algorithm that assigns a score to each semantic triple, indicating the reliability of the information based on the data source. This scoring system is crucial for security researchers to ascertain the accuracy of recorded cyber-events and draw parallels with known incidents. While this aspect is not addressed in our present work, it remains a significant consideration, which we have earmarked for enhancement in future iterations.

Rahman and Williams [32]'s study on extracting Attack Techniques from Unstructured Text compares various techniques for extracting attack techniques from unstructured text, such as threat reports. The paper identifies Term Frequency-Inverse Document Frequency (TFIDF) and Latent Semantic Indexing (LSI) as the most effective methods, achieving F1 scores of 84% and 83%, respectively. The research also explores the impact of class label increase on method performance and suggests oversampling as a strategy to address class imbalance issues. This paper presents various existing approaches to attack technique extraction, we take inspiration from it and explore an alternative method that we believe could also be a viable option.

In the area of using Knowledge Graphs for associating MITRE ATT&CK techniques with CTI reports, Kriaa and Chaabane [13] presents a novel approach to leveraging knowledge graphs for attack detection and prediction. The primary contributions of this paper revolve around the development of the SecKG schema, the detection module, and the prediction module. These components facilitate learning from a KG, thereby enhancing the system's ability to foresee attacks. One of the major issues highlighted is the high rate of false positives generated by the prediction module due to the scarcity of instances of specific attack data in the training dataset, and the model needs to be optimized further. However, the rate of prediction is quite promising even with the limited amount of data.

Liu and Zhan [33] employs ChatGPT[22] to identify and categorize attack-related elements and their interconnections for efficiently constructing knowledge graphs from CTI. It shows potential in low-resource scenarios, showcasing better results as compared to AttackKG and REBEL, although the model still requires further calibration and validation, since, to quote the author, *"ChatGPT, while an impressive tool, is occasionally too "creative" to create relations that might be incorrect"*.

The body of literature reviewed here provides an overview of the current state of research in the application of LLMs on graphs, their use in cybersecurity, the role of CKGs, and the extraction of attack techniques from unstructured text. Each of these studies contributes valuable insights and methodologies that have informed our approach.

Below sections expand on the models and tools that form the foundation of our work, where we preprocess and enrich the input data from TRAM (Section 2.3.3) with information from MITRE ATT&CK and train models extended from KEPLER (Section 2.3.4) which is built on top of FAIRSEQ (Section 2.3.1) and RoBERTa (Section 2.3.2). The trained model is then converted to Hugging Face Transformer and further fine-tuned and finally evaluated against the TRAM(Section 2.3.3) model.

2.3 Models & Tools

2.3.1 FAIRSEQ

The Facebook AI Research Sequence-to-Sequence (FAIRSEQ)[23] model, developed by Fundamental AI Research (FAIR) team at Meta[24], represents an

[22]https://chat.openai.com/
[23]https://fairseq.readthedocs.io/en/latest/
[24]https://ai.meta.com/research/

advancement in the field of sequence modeling. While earlier models prioritized either extensibility or performance, FAIRSEQ stands out for its speed, extensibility, and versatility, making it suitable for both research and production environments [34]. This model is part of the FAIRSEQ toolkit, which is an open-source platform designed for a variety of text generation tasks such as translation, summarization, and language modeling. The FAIRSEQ toolkit, and by extension the FAIRSEQ model, is built upon the PyTorch framework and supports distributed training across multiple GPUs and machines, facilitating rapid and efficient model development.

FAIRSEQ's architecture is engineered to be robust, addressing the challenges of real-world conditions that are often not represented in public datasets. The FAIRSEQ model aims to bridge this gap by providing a benchmark that matches real-life conditions, thereby enabling the identification of the strengths and weaknesses of the models more effectively.

2.3.2 RoBERTa

The advent of BERT marked a paradigm shift in the field of NLP, setting new standards for a variety of tasks [19]. However, subsequent research by Liu et al. [21] introduced A Robustly Optimized BERT Pretraining Approach (RoBERTa), and demonstrated that the original BERT model was significantly under-trained.

RoBERTa revisits the BERT pretraining methodology and scales it up, leading to improved performance across a range of NLP benchmarks. The study conducted by Liu et al. [21] measured the impact of key hyperparameters and training data size, which were previously overlooked or not optimized in the original BERT pretraining. The findings revealed that with careful tuning and extended training on larger datasets, RoBERTa could match or even surpass the performance of all models published after BERT, including those claiming significant improvements over it.

- **Training on Larger Datasets** : RoBERTa was trained on a dataset comprising 160 GB of text, which is ten times larger than the dataset used for BERT.

- **Removing BERT's Next Sentence Prediction (NSP)** : The NSP task was found to be an ineffective pretraining objective, and its removal led to performance gains.

- **Dynamic Masking** : Unlike BERT's static masking, where the masked tokens are predetermined before training starts, RoBERTa applies dynamic masking, generating the masking pattern every time a sequence is fed into the model.
- **Longer Training with Larger Batches** : RoBERTa extends the training process, using larger batch sizes and more training steps, which contributes to its robustness and effectiveness.

The RoBERTa model has set new state-of-the-art results on major NLP benchmarks such as General Language Understanding Evaluation (GLUE) [35], Reading Comprehension Dataset From Examinations (RACE) [36], and Stanford Question Answering Dataset (SQuAD) [37], highlighting the significance of revisiting and refining pretraining approaches in the development of NLP models

2.3.3 TRAM

The Threat Report ATT&CK Mapper (TRAM)[25] is an open-source platform developed by MITRE Engenuity[26]. It is designed to address the challenges faced by the cybersecurity community in identifying TTPs in CTI reports.

Large Language Models The most recent version of this initiative has enhanced the training data's quality and leveraged fine-tuned LLMs to boost the efficiency of model training and forecasting.

Training the LLM The pretrained SciBERT LLM is further trained on the annotated data from the TRAM tool. This allows the model to learn the patterns and relationships between the text in the CTI reports and the corresponding TTPs.

Predicting TTPs Once the LLM is trained, it can be used to automatically identify TTPs in new CTI reports. The model analyses the text in the report and predicts the TTPs based on what it has learned during training.

[25] `https://mitre-engenuity.org/cybersecurity/center-for-threat-informed-defense/our-work/threat-report-attck-mapper-tram/`

[26] `https://mitre-engenuity.org/`

2.3.4 KEPLER

Knowledge Embedding and Pre-trained LanguagE Representation (KEPLER) [14] is a model that sits at the intersection of KG and PLMs, aiming to capture factual knowledge from both structured KGs and unstructured text.

It is built upon the FAIRSEQ(Section 2.3.1) framework, leveraging its robust foundation. This design choice allows KEPLER to benefit from FAIRSEQ's efficient training and generation capabilities, as well as its extensibility. Notably, it also maintains compatibility with RoBERTa's (Section 2.3.2) model architecture. As a result, KEPLER checkpoints can be utilized in the same manner as RoBERTa checkpoints for various downstream NLP tasks.

Model Architecture KEPLER encodes textual entity descriptions using a PLM to generate entity embeddings, which are then optimized jointly with the knowledge embedding and language modeling objectives. This joint optimization allows KEPLER to integrate factual knowledge into PLMs more effectively and produce text-enhanced KE with strong PLMs. The model architecture (Figure 2.4) is designed to be robust, handling the sparse and complex nature of world facts as they appear in the text.

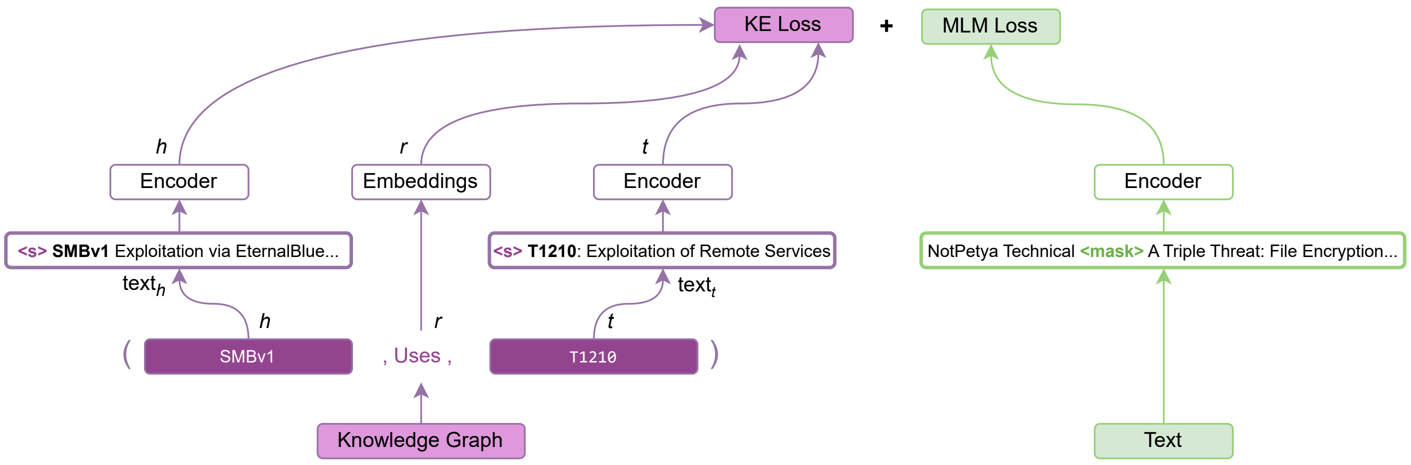
KE Loss
+
MLM Loss
h
r
t
Encoder
Embeddings
Encoder
Encoder
<s> SMBv1 Exploitation via EternalBlue...
<s> T1210: Exploitation of Remote Services
NotPetya Technical <mask> A Triple Threat: File Encryption...
text_h
text_t
h
r
t
(SMBv1 , Uses , T1210)
Knowledge Graph
Text

Figure 2.4: The KEPLER framework. Entity descriptions are encoded as entity embeddings and the KE and MLM objectives are jointly trained on the same PLM [14].

Dataset and Pre-training The researchers developed Wikidata5M[27] for pre-training and evaluating KEPLER, a comprehensive KG dataset featuring aligned entity descriptions. This dataset establishes a benchmark for KE methods and supports research on expansive KGs, inductive KE, and KGs enriched with textual data. KEPLER's pre-training involves encoding the textual descriptions of entities and then optimizing the model to predict the relationships between these entities accurately.

Features :

- **Textual Entity Descriptions** : Entities are described using text, which provides context and improves the quality of the embeddings.
- **Joint Optimization** : The model is trained with a joint loss function that combines the objectives of KE and MLM, leading to a more holistic learning process.
- **Inductive Capabilities** : KEPLER functions well as an inductive KE model, effectively handling KG link prediction tasks even for unseen data.

Experimental Results The experimental findings reveal that KEPLER sets a new standard in performance across diverse NLP tasks, one example of which has been detailed below.

[27] https://deepgraphlearning.github.io/project/wikidata5m

Model	FewRel 1.0				FewRel 2.0			
	5-1	5-5	10-1	10-5	5-1	5-5	10-1	10-5
MTB ($\text{BERT}_{\text{LARGE}}$)†	93.86	97.06	89.20	94.27	–	–	–	–
Proto (BERT)	80.68	89.60	71.48	82.89	40.12	51.50	26.45	36.93
Proto (MTB)	81.39	91.05	71.55	83.47	52.13	76.67	48.28	69.75
Proto ($\text{ERNIE}_{\text{BERT}}$)†	89.43	94.66	84.23	90.83	49.40	65.55	34.99	49.68
Proto ($\text{KnowBert}_{\text{BERT}}$)†	86.64	93.22	79.52	88.35	64.40	79.87	51.66	69.71
Proto (RoBERTa)	85.78	95.78	77.65	92.26	64.65	82.76	50.80	71.84
Proto (Our RoBERTa)	84.42	95.30	76.43	91.74	61.98	83.11	48.56	72.19
Proto ($\text{ERNIE}_{\text{RoBERTa}}$)†	87.76	95.62	80.14	91.47	54.43	80.48	37.97	66.26
Proto ($\text{KnowBert}_{\text{RoBERTa}}$)†	82.39	93.62	76.21	88.57	55.68	71.82	41.90	58.55
Proto (KEPLER-Wiki)	88.30	95.94	81.10	92.67	66.41	84.02	51.85	73.60
PAIR (BERT)	88.32	93.22	80.63	87.02	67.41	78.57	54.89	66.85
PAIR (MTB)	83.01	87.64	73.42	78.47	46.18	70.50	36.92	55.17
PAIR ($\text{ERNIE}_{\text{BERT}}$)†	92.53	94.27	87.08	89.13	56.18	68.97	43.40	54.35
PAIR ($\text{KnowBert}_{\text{BERT}}$)†	88.48	92.75	82.57	86.18	66.05	77.88	50.86	67.19
PAIR (RoBERTa)	89.32	93.70	82.49	88.43	66.78	81.84	53.99	70.85
PAIR (Our RoBERTa)	89.26	93.71	83.32	89.02	63.22	77.66	49.28	65.97
PAIR ($\text{ERNIE}_{\text{RoBERTa}}$)†	87.46	94.11	81.68	87.83	59.29	72.91	48.51	60.26
PAIR ($\text{KnowBert}_{\text{RoBERTa}}$)†	85.05	91.34	76.04	85.25	50.68	66.04	37.10	51.13
PAIR (KEPLER-Wiki)	90.31	94.28	85.48	90.51	67.23	82.09	54.32	71.01

Table 2.2: The table taken from KEPLER [14] paper shows ***Accuracies*** *(%) on the FewRel dataset [38]. N-K indicates the N-way K-shot setting. "Proto" indicates Prototypical Networks [39], "PAIR" is from Gao et al. [40] and "MTB" is from Soares et al. [41]. MTB uses the* ***LARGE*** *size and all the other models use the* ***BASE*** *size. † indicates oracle models which may have seen facts in the FewRel 1.0 test set during pre-training.*

In the evaluation shown below (Table 2.2), several models classify queries into relations based on sampled instances. FewRel[28] and FewRel 2.0[29] are few-shot relation classification datasets that were used here. Two frameworks, Proto [39] and PAIR [40] were used where the text encoders were replaced by KEPLER and the other baseline models used in the paper. KEPLER-Wiki outperformed other BASE-size PLM in most settings. Interestingly, RoBERTa-based models were comparable or even inferior to BERT-based models in the PAIR framework. KEPLER showed improvements on FewRel 2.0, while ERNIE and KnowBert declined in most settings. This suggests that KEPLER not only learns better entity representations but also demon-

[28]https://paperswithcode.com/dataset/fewrel
[29]https://paperswithcode.com/dataset/fewrel-2-0

strates a general ability to extract factual knowledge from the context across different domains. Conversely, ERNIE and KnowBert may not generalize well to new domains requiring different entity linkers and entity embeddings. Additionally, KEPLER exhibits good inductive KE proficiency in KG link prediction, surpassing previous methodologies.

Model	MR	MRR	HITS@1	HITS@3	HITS@10
DKRL [42]	78	23.1	5.9	32.0	54.6
RoBERTa	723	7.4	0.7	1.0	19.6
Our RoBERTa	1070	5.8	1.9	6.3	13.0
KEPLER-KE	138	17.8	5.7	22.9	40.7
KEPLER-Rel	35	33.4	15.9	43.5	66.1
KEPLER-Wiki	32	35.1	15.4	46.9	71.9
KEPLER-Cond	28	40.2	22.2	51.4	73.0

Table 2.3: Table taken from KEPLER [14] paper shows Link prediction results on Wikidata5M (% except MR) inductive settings. The evaluation metrics are Mean reciprocal rank (MRR), Mean rank (MR), and HITS@K (which refers to the number of actual positive triples that are positioned within the top-k ranks when compared to a set of artificially created negative triples).

The inductive prediction results in Table 2.3 of the KEPLER model, as evaluated on Wikidata5M, demonstrate a significant improvement over the DKRL [42] and RoBERTa models. This is indicative of the effectiveness of the joint training method employed by KEPLER. However, the paper also states that the performance of KEPLER, while superior, is still not at the level required for practical applications such as constructing a knowledge graph from scratch. This shows the need for further advancements in inductive KE.

Chapter 3

Materials & Methods

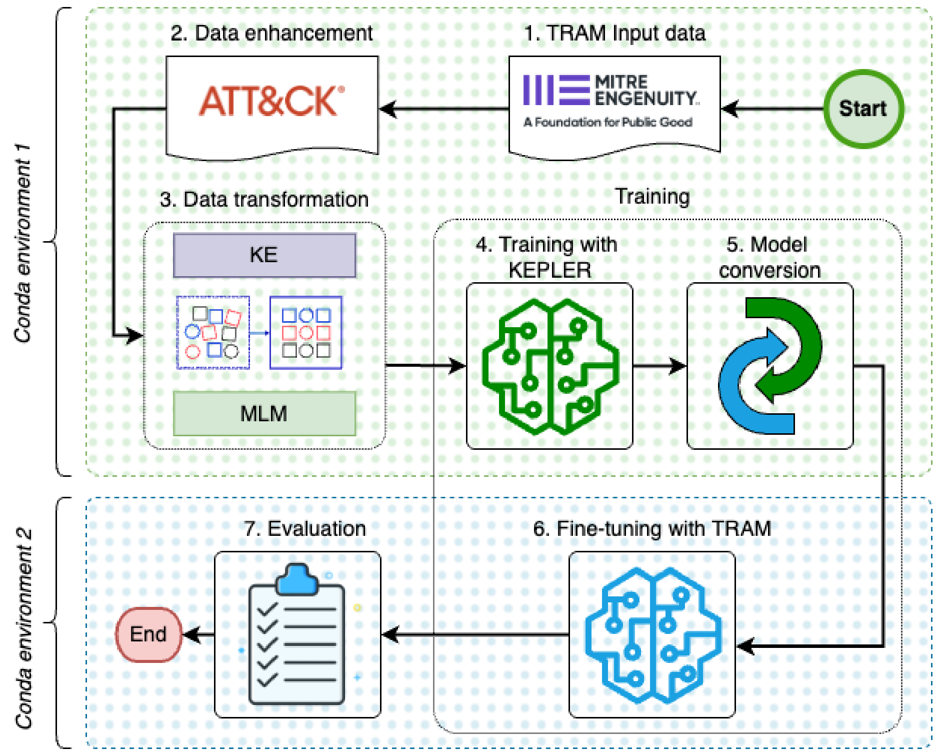

Figure 3.1: This figure shows the method pipeline that is followed in the thesis work.

3.1 Data Identification

Our work focuses on data from **MITRE ATT&CK**, which is a free-to-use and comprehensive Knowledge Base of real-world data on tactics, techniques, and procedures used by cyber adversaries. It offers valuable insights into the various stages of an attack, from initial reconnaissance to actions on objectives, and is used to build threat models and methodologies in various sectors, including private companies, government, and cybersecurity services.

The data is organized into a matrix layout[30] that represents the various stages of an attack. This matrix is divided into several categories, each representing a different phase of a cyberattack:

The April 2024 (v15) ATT&CK release[31] for the Enterprise dataset contains 14 Tactics, 202 Techniques, 435 Sub-Techniques, 148 Groups, 677 Pieces-of-Software, 28 Campaigns, 43 Mitigations, and 37 Data-Sources.

A subset of the MITRE ATT&CK data, which has been used in the most recent **TRAM2** annotation initiative[32], was used as an input to our work, as the first step of the pipeline (Figure 3.1). This is an annotated information that covers 50 ATT&CK techniques, and is stored in two datasets:[33]

- `single_label.json` has annotations at the phrase level. Due to the current data limitations, experiments with this dataset show better outcomes with TRAM's prediction algorithm. An example entry of the file looks like the Listing 3.2.2

- `multi_label.json` provides annotations at the sentence level. This approach is more versatile, allowing the model to identify references to various techniques in closely situated text, but prediction accuracy with this is worse due to the small size of the dataset. An example of this is as follows:

[30] https://attack.mitre.org

[31] https://attack.mitre.org/resources/updates/updates-april-2024/

[32] https://github.com/center-for-threat-informed-defense/tram/wiki/Data-Annotation

[33] https://github.com/center-for-threat-informed-defense/tram/tree/main/data/tram2-data

```
{
 "sentence": "Once the victim opens the document, the embedded
    macro will be executed, injecting the shellcode into rundll32.
    exe.",
 "labels": [
    "T1055",
    "T1204.002"
 ],
 "doc_title": "Earth Zhulong Familiar Patterns Target Southeast
    Asian Firms"
}
```

The following sections explain how the single-labeled input data was pre-processed, trained, and tested. The same steps apply to multi-labeled input data, with some minor changes in the algorithm that create the triples.

3.2 Data Pre-processing

3.2.1 Environment Setup

Initially, a Conda[34] environment was established with the following specifications. This was done to make sure that the KEPLER code works as intended:

- PyTorch 1 [35]
- Python 3.9 [36]
- NVIDIA Apex libraries [37]
- KEPLER's installation from source, which also included a customized FAIRSEQ 0.8.0 source code [38]
- Hugging Face Transformers 2 [39]

[34] https://conda.io/
[35] https://pytorch.org/get-started/previous-versions/#v1131
[36] https://www.python.org/downloads/release/python-390/
[37] https://github.com/NVIDIA/apex
[38] https://github.com/SOUGATA/KEPLER
[39] https://huggingface.co/transformers/v2.3.0/index.html

3.2.2 Data Enrichment

The annotated TRAM single-label dataset was converted into a tabular format. Additionally, Tactic IDs, Tactic names, Technique names, and procedure examples corresponding to the Techniques, were extracted and added[40] from MITRE ATT&CK[41] to this dataset. This is the second step in the pipeline. (Figure 3.1)

As an example, the following data from TRAM:

```
{
 "text": "elevate its privileges by executing the mw_UAC_bypass
    function",
 "label": "T1548.002",
 "doc_title": "Enigma Stealer Targets Cryptocurrency Industry
    with Fake Jobs"
}
```

has been enriched with related Tactic IDs, Tactic names, and Procedure examples from MITRE ATT&CK dataset as shown below.

The name and Id of the Tactic node:

```
{
    "x_mitre_domains": ["enterprise-attack"],
    "id": "x-mitre-tactic--5e29b093-294e-49e9-a803-dab3d73b77dd
       ",
    "type": "x-mitre-tactic",
    "external_references": [
        {
            "external_id": " TA0004 ",
            "url": "https://attack.mitre.org/tactics/TA0004",
            "source_name": "mitre-attack"
        }
    ],
    "name": " Privilege Escalation ", ...
}
```

[40] https://github.com/SOUGATA/KEPLER/blob/feature/TRAM/tram2kepler/scripts/singleLabel2kepler_aug.ipynb
[41] https://github.com/mitre-attack/attack-stix-data/blob/master/enterprise-attack/enterprise-attack.json

The name corresponding to the Technique node:

```
{
    "name": " Bypass User Account Control ",
    "kill_chain_phases": [
        {
            "kill_chain_name": "mitre-attack",
            "phase_name": "privilege-escalation"
        }, ...
    ],
    "x_mitre_is_subtechnique": true,
    "type": "attack-pattern",
    "id": "attack-pattern--120d5519-3098-4e1c-9191-2aa61232f073
        ",
    "external_references": [
        {
            "source_name": "mitre-attack",
            "url": "https://attack.mitre.org/techniques/T1548
                /002",
            "external_id": "T1548.002"
        }, ...
    ], ...
}
```

The Procedure Example name & external_Id from its corresponding node:

```
{
    "name": " APT37 ",
    "type": "intrusion-set",
    "id": "intrusion-set--4a2ce82e-1a74-468a-a6fb-bbead541383c",
    "external_references": [
        {
            "source_name": "mitre-attack",
            "url": "https://attack.mitre.org/groups/G0067",
            "external_id": " G0067 "
        },
        ...
    ],
   ...
}
```

3.2.3 Data Transformation

The enriched dataset, containing Technique IDs, CTI Document Titles, text snippets, Tactic IDs, Tactic names, Technique names, and Procedure Example names, formed the basis of the entity descriptions set and was subsequently used for generating inputs for both the KE and MLM objectives. This forms the third step of the pipeline. (Figure 3.1)

3.2.3.1 Knowledge Embedding

For the KE objective, a set of triples was to be generated, as shown in Figure 2.3. Each relationship type was assigned a numeric value, and the subject and object were assigned their corresponding index, in the form of their line numbers in the entity-descriptions set. By iterating over the original tabular data and determining their corresponding indexes in the entity-descriptions set, the set of triples was created. The triples set and the entity-descriptions set were subsequently divided into training, validation, and test subsets.[42]

Extending on the approach in KEPLER, the entity-descriptions set was encoded using GPT2 BPE(Section 2.1.1.3). [43]

Negative sampling was then performed, and then the encoded-entity-descriptions set was divided into training and validation samples. [44]

The next step was to binarize the training, validation, and negative samples. [45]

3.2.3.2 Masked Language Model

Similar to the KE preprocessing steps, the training, validation and test subsets of the entity-descriptions set were encoded using a GPT2 encoder[46]. The next step was to binarize the encoded data using the GPT2 dictionary[47].

[42] See footnote 40

[43] https://github.com/SOUGATA/KEPLER/blob/main/examples/roberta/multiprocessing_bpe_encoder.py

[44] https://github.com/SOUGATA/KEPLER/blob/main/examples/KEPLER/Pretrain/KGpreprocess.py

[45] https://github.com/SOUGATA/KEPLER/blob/feature/TRAM/preprocess.py

[46] https://github.com/SOUGATA/KEPLER/blob/main/examples/roberta/multiprocessing_bpe_encoder.py

[47] https://github.com/SOUGATA/KEPLER/blob/main/preprocess.py

3.2.3.3 Trial with PyTorch Geometric

Data transformation to a Pytorch Geometric (PyG) HeteroData[48] KG was also carried out from the TRAM dataset. Since extensive code changes were to be done if PyG were to be used with KEPLER, this has been kept as a future enhancement candidate.

3.3 Training

3.3.1 Training with KEPLER

A pre-trained KEPLER NLP checkpoint[49] was used as the starting point for further training[50] it on both the pre-processed KE and MLM data, keeping the hyperparameters[51], other than the number of updates, unchanged as in the KEPLER's documentation.[52]. This is the fourth step in the pipeline. (Figure 3.1)

3.3.2 Checkpoint Conversion

The best model checkpoint, which was selected by the KEPLER's training process, was then selected. This KEPLER checkpoint was then converted[53] to a Hugging Face transformer checkpoint[54] for ease of further processing. This was done on the first Conda environment due to incompatibility between the FAIRSEQ versions used by KEPLER and the latest Hugging Face Transformers. This is the fifth step in the pipeline. (Figure 3.1)

[48] https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.data.HeteroData.html#torch-geometric-data-heterodata

[49] https://cloud.tsinghua.edu.cn/f/e03f7a904526498c81a4/

[50] https://github.com/SOUGATA/KEPLER/blob/feature/TRAM/train.py

[51] https://github.com/SOUGATA/KEPLER/blob/feature/TRAM/tram2kepler/scripts/singleLabel2kepler_v2.ipynb

[52] https://github.com/SOUGATA/KEPLER/blob/main/README.md#running

[53] https://github.com/SOUGATA/KEPLER/blob/feature/TRAM/tram2kepler/scripts/kepler2hft.sh

[54] https://github.com/huggingface/transformers

3.3.3 Second Environment Setup

A second Conda environment was set up with the following specifications. This was done to make sure that the newer TRAM training and evaluation works as intended:

- PyTorch 2.0 [55]
- Python 3.12 [56]
- Hugging Face Transformers 4 [57]

3.3.4 Fine-tuning with TRAM

The Hugging Face Transformers model was loaded and fine-tuned[58] with a modified version of the TRAM script.[59]

The two major modifications were that the model thus loaded was now a `GPT2ForSequenceClassification`[60], whereas the TRAM algorithm uses `BertForSequenceClassification`[61], and that the last layer of the model was now set to be of 50 class outcomes, to match with the TRAM goal of predicting amongst 50 Techniques.

This model was then fine-tuned on the TRAM data, to learn new weights to be able to predict amongst 50 class labels.

This is the sixth step in the pipeline. (Figure 3.1)

[55] https://pytorch.org/get-started/pytorch-2.0/
[56] https://www.python.org/downloads/release/python-3123/
[57] https://huggingface.co/docs/transformers/index
[58] https://github.com/SOUGATA/KEPLER/blob/feature/TRAM/tram2kepler/scripts/tram/fine_tune_single_label.py
[59] https://github.com/center-for-threat-informed-defense/tram/blob/main/user_notebooks/fine_tune_single_label.ipynb
[60] https://huggingface.co/docs/transformers/v4.40.1/en/model_doc/gpt2#transformers.GPT2ForSequenceClassification
[61] https://huggingface.co/docs/transformers/v4.40.1/en/model_doc/bert#transformers.BertForSequenceClassification

3.4 Evaluation

Finally, the evaluation was done with a CTI report named `Enigma Stealer Targets Cryptocurrency Industry with Fake Jobs`[62], converted to a PDF[63], to mimic a real-life scenario. Parts of this report are also present in the original TRAM input datasets. This is the seventh and final step in the pipeline. (Figure 3.1)

The file was uploaded to a program that split up the text as per the configured length and stride hyperparameters (Default values from the TRAM script were used: length=**13**, stride=**5**, so that the performance could be measured against TRAM effectively), while the outcomes were noted with a range of prediction confidence threshold values, to observe how accurately it predicts techniques according to the text descriptions from the CTI report. In the Results section, prediction outcome details are given for the confidence threshold of **0.9**, which is also the standard value that TRAM uses in its reports.

3.5 Software & Hardware Specifications

- Two Conda environments (Sections 3.2.1 and 3.3.3) were set up to work with KEPLER and TRAM respectively.
- Overleaf[64] was used as the collaboration platform for thesis writing, and Github[65] was used for the codebase.
- LanguageTool[66] and Grammarly[67] were used for checking grammatical inaccuracies in the thesis text.
- Matplotlib library[68] has been used to generate graphs.

[62] `https://www.trendmicro.com/en_nl/research/23/b/enigma-stealer-targets-cryptocurrency-industry-with-fake-jobs.html`
[63] `https://github.com/SOUGATA/KEPLER/blob/feature/TRAM/tram2kepler/data/input/Enigma%20Stealer%20Targets%20Cryptocurrency%20Industry%20with%20Fake%20Jobs%20_%20Trend%20Micro.pdf`
[64] `https://www.overleaf.com/`
[65] `https://github.com/SOUGATA/KEPLER/tree/feature/TRAM`
[66] `https://languagetool.org/`
[67] `https://app.grammarly.com/`
[68] `https://matplotlib.org/`

- draw.io[69] has been used to create diagrams.
- A Macbook Pro (Chip: Apple M3 Max, macOS: 14.4.1 (23E224)) was used to write the required code and the thesis.
- Resource intensive tasks like model training, fine-tuning, and evaluation were run on:
  - Google Colab: Tesla T4 GPU, System RAM 12.7GB; GPU Ram: 15GB, Disk 78.2 GB [70]
  - SIMULA: eX^3 partition dgx2q: node g001: Nvidia DGX-2 Dual Processor Xeon Scalable 8168 48cores and 1.5TB RAM and 30TB NVMe

3.6 Usage of AI in thesis writing

Microsoft Copilot[71], which uses OpenAI's language model (GPT-4), has been used to rephrase and refine some paragraphs of this thesis. The prompts that were used were similar to: *"Rephrase as an academic text: ...a paragraph..."*. However, the outcomes were always checked for inaccuracies or exaggerations and edited before final use.

[69] https://draw.io/
[70] https://colab.research.google.com/
[71] https://copilot.microsoft.com/

Chapter 4

Results

In this section, the outcomes of data preparation and transformation, along with training and evaluation results, are presented. All data shown here are saved in a GitHub repository[72] for reference.

4.1 Data Dimensions

4.1.1 Base data

The following tables (4.1, 4.4) present the item counts that were extracted from the TRAM input data. The nodes are counts of items represented in the input files, and the generation of triples was based on the relationships between these nodes, adhering to our established ontology (Figure 2.2).

[72] https://github.com/SOUGATA/KEPLER/tree/feature/TRAM/tram2kepler/scripts/results

4.1.1.1 Single-labeled

Nodes	#	Triples	#
Technique Ids	50	Texts - TechniqueId	5089
Doc Titles	149	Texts - Doc Titles	5089
Texts	5089	TechniqueIds - Doc Titles	1690
Total	5288	Total	11868

Table 4.1: Dimensions for single-labeled TRAM data.

The KEPLER model that was trained on the single-labeled base data (Table 4.1), has **5288** lines of encoded text as an input to its MLM objective, and a Knowledge Graph comprising **5288** nodes, with **11868** relationship triples connecting those nodes, was the input to its KE objective.

The next two figures (4.1, 4.2) show the frequency of techniques and document titles associated with the text excerpts from the CTI reports, in the single-labeled dataset. The imbalance in the distribution of these data is because the TRAM input data comprises the **50** most commonly found techniques[73] from a selected set of **149** CTI reports.

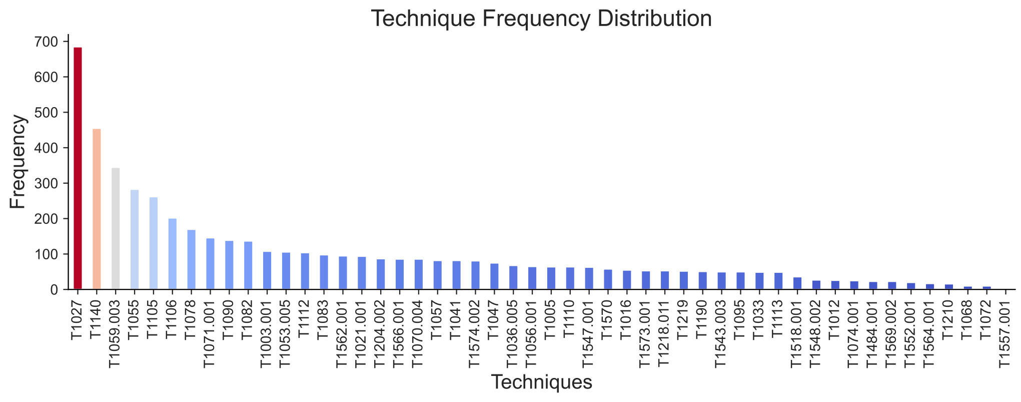

Figure 4.1: Frequency of Techniques associated with text excerpts present in the single-labeled dataset.

[73]https://github.com/center-for-threat-informed-defense/tram/wiki/Large-Language-Models#subset-of-techniques

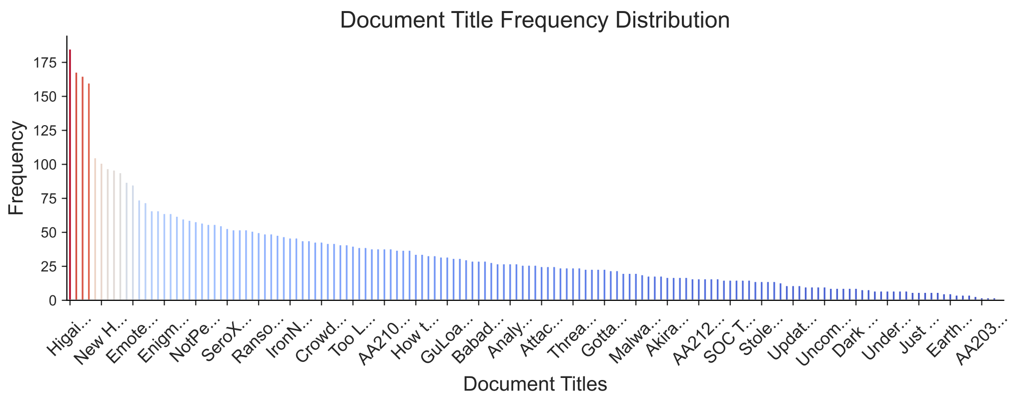

Figure 4.2: Frequency of Document titles associated with text excerpts present in the single-labeled dataset.

4.1.1.2 Multi-labeled

Nodes	#
TechniqueIds	50
Doc Titles	151
Sentences	19178
Total	19379

Triples	#
Sentences - TechniqueIds	5143
Sentences - Doc Titles	19178
TechniqueIds - Doc Titles	1653
Total	25974

Table 4.2: Dimensions for multi-labeled TRAM data.

The KEPLER model trained on multi-labeled data (Table 4.2) shows that **19379** lines of encoded texts were used for the MLM objective, while **19379** nodes with **25974** relationship triples pointing amongst those nodes were used as input for training the KE objective.

The next two figures (4.3, 4.4) show the frequency of techniques and document titles associated with the phrases from the CTI reports, in the multi-labeled dataset. The imbalance in distribution of these data is because the TRAM input data comprises the **50** most commonly found techniques[74] in a selected set of **151** CTI reports.

[74]https://github.com/center-for-threat-informed-defense/tram/wiki/Large-Language-Models#subset-of-techniques

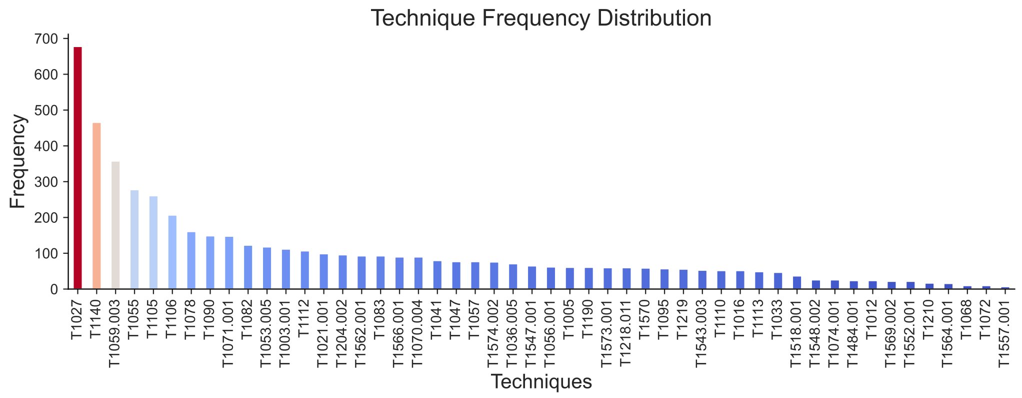

Figure 4.3: Frequency of Techniques present in the multi-labeled dataset.

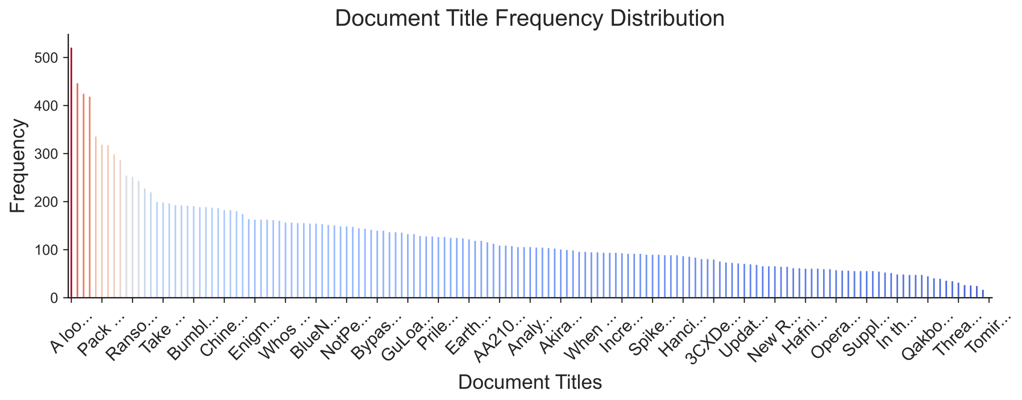

Figure 4.4: Frequency of Document titles present in the multi-labeled dataset.

4.1.2 Enriched data

The next two tables show information about the datasets that were generated based on single and multi-labeled data (section 4.1), in conjunction with enriched information, such as technique names, tactic identifiers and names, and procedure examples, derived from the MITRE ATT&CK dataset, as detailed in the section that deals with Data Preprocessing (3.2). The enriched nodes and triples are highlighted in yellow.

4.1.2.1 Single-labeled

Nodes	#
TechniqueIds	50
TacticIds	11
Technique Names	50
Tactic Names	11
Doc Titles	149
Texts	5089
Procedures	814
Total	6174

Triples	#
Texts - TechniqueIds	5089
Doc Titles - Texts	5089
TechniqueIds - Doc Titles	1690
TechniqueIds - TacticIds	50
TechniqueNames - TechniqueIds	50
TacticNames - TacticIds	11
Procedures - TechniqueIds	6833
Total	18812

Table 4.3: Dimensions for single-labeled TRAM data enriched with MITRE ATT&CK information.

4.1.2.2 Multi-labeled

Nodes	#
TechniqueIds	50
TacticIds	11
Technique Names	50
Tactic Names	11
Doc Titles	151
Sentences	19178
Procedures	814
Total	20265

Triples	#
Sentences - TechniqueIds	5143
Doc Titles - Sentences	19178
TechniqueIds - Doc Titles	1653
TechniqueIds - TacticIds	50
TechniqueNames - TechniqueIds	50
TacticNames - TacticIds	11
Procedures - TechniqueIds	6833
Total	32918

Table 4.4: Dimensions for multi-labeled TRAM data enriched with MITRE ATT&CK information.

4.2 Model Structures

4.2.1 KEPLER model based on RoBERTa

This shows the structure of the KEPLER model that was used for initial training on the MLM and KE objectives.

```
RobertaModel(
  (decoder): RobertaEncoder(
    (sentence_encoder): TransformerSentenceEncoder(
      (embed_tokens): Embedding(50265, 768, padding_idx=1)
      (embed_positions): LearnedPositionalEmbedding(514, 768,
          padding_idx=1)
      (layers): ModuleList(
        (0-11): 12 x TransformerSentenceEncoderLayer(
          (self_attn): MultiheadAttention(
            (out_proj): Linear(in_features=768, out_features=768,
                bias=True))
          (self_attn_layer_norm): LayerNorm((768,), eps=1e-05,
              elementwise_affine=True)
          (fc1): Linear(in_features=768, out_features=3072, bias=
              True)
          (fc2): Linear(in_features=3072, out_features=768, bias=
              True)
          (final_layer_norm): LayerNorm((768,), eps=1e-05,
              elementwise_affine=True)))
      (emb_layer_norm): LayerNorm((768,), eps=1e-05,
          elementwise_affine=True))
    (lm_head): RobertaLMHead(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (layer_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=
          True)))
  (classification_heads): ModuleDict()
  (ke_heads): ModuleDict(
    (wikiData): RobertaKnowledgeEmbeddingHead(
      (relation_emb): Embedding(822, 768)
    )))
```

4.2.2 KEPLER model based on Hugging Face transformers

This was the structure of the KEPLER model after conversion to Hugging Face transformer, which was finally used for the prediction of labels.

```
GPT2ForSequenceClassification(
  (transformer): GPT2Model(
    (wte): Embedding(50265, 768)
    (wpe): Embedding(514, 768)
    (drop): Dropout(p=0.1, inplace=False)
    (h): ModuleList(
      (0-11): 12 x GPT2Block(
        (ln_1): LayerNorm((768,), eps=1e-05, elementwise_affine=True
            )
        (attn): GPT2Attention(
          (c_attn): Conv1D()
          (c_proj): Conv1D()
          (attn_dropout): Dropout(p=0.1, inplace=False)
          (resid_dropout): Dropout(p=0.1, inplace=False)
        )
        (ln_2): LayerNorm((768,), eps=1e-05, elementwise_affine=True
            )
        (mlp): GPT2MLP(
          (c_fc): Conv1D()
          (c_proj): Conv1D()
          (act): NewGELUActivation()
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
    (ln_f): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
  )
  (score): Linear(in_features=768, out_features=50, bias=False)
)
```

4.2.3 TRAM model

The TRAM model structure is given here as a reference to what the above model was evaluated against. As can be seen, this is a deeper model with more layers than the KEPLER models above, which usually relates to the ability to learn more diverse and complex patterns.

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(31090, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False))
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False))
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=
                  True)
              (dropout): Dropout(p=0.1, inplace=False)))
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation())
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=
                True)
            (dropout): Dropout(p=0.1, inplace=False)))))
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()))
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=50, bias=True))
```

4.3 Training Outcomes

The graphs in the following four subsections depict the change in training and validation losses, and perplexity throughout the epochs. Each subsection refers to a specific dataset the model was trained on.

Each set of graphs is arranged as follows: Training-VS-Validation Loss, Training Loss, Validation Loss and Perplexity, over the epochs.

4.3.1 Single-labeled data

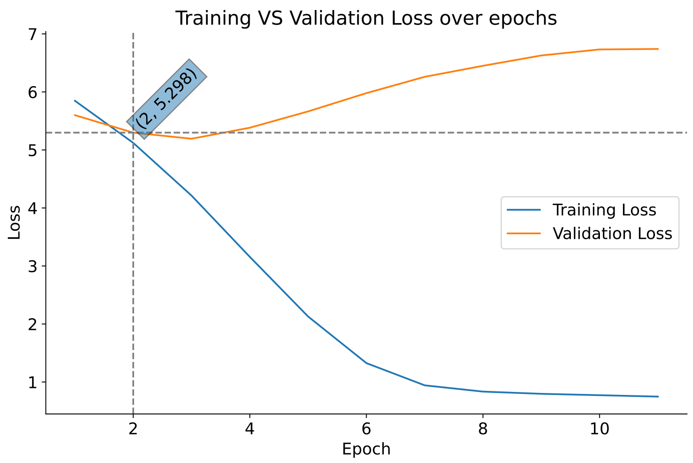

Figure 4.5: Training VS Validation loss for the single-labeled dataset.

It can be seen that the training and validation *Total loss*es cross quickly at epoch 2, after which the model overfits, showing that the model is capable of learning the weights at a very early iteration of the training process (Figure 4.5).

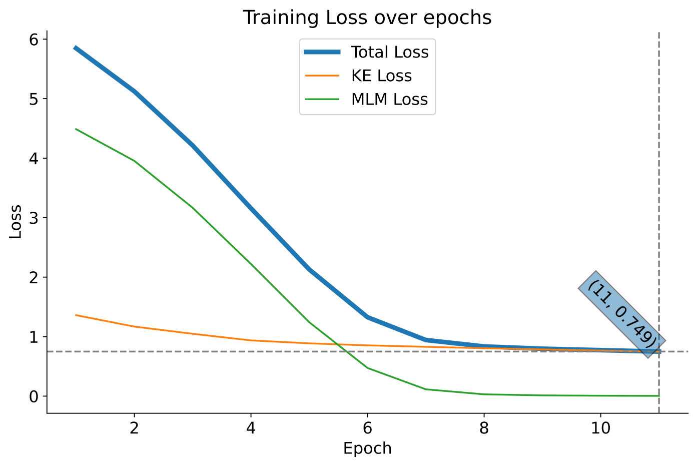

Figure 4.6: Training loss for single-labeled dataset.

In the training loss graph (Figure 4.6), the thicker *Total loss* line is the value on which the model is trained. It is the sum of MLM and KE losses. It is observed that the MLM loss decreases within the first 10 epochs before flattening out. KE losses start at a lower value and decrease slightly over the epochs.

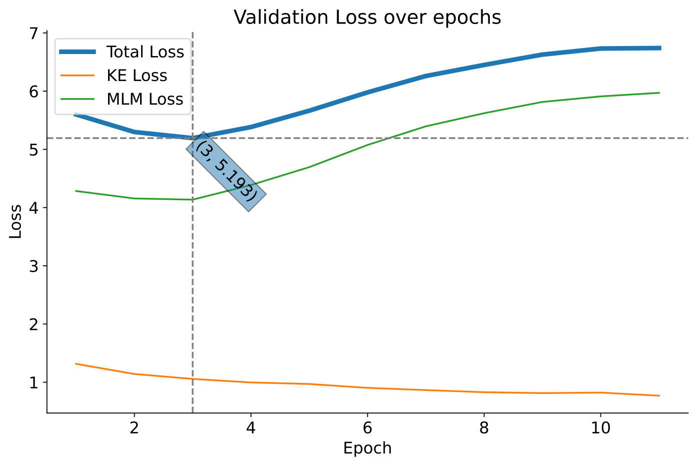

Figure 4.7: Validation loss for the single-labeled dataset.

In the validation loss graph (Figure 4.7), the thicker *Total loss* line is the value against which the model is validated. It is the sum of MLM and KE losses. The MLM loss decreases until the 3rd epoch before increasing continuously, showing classic signs of overfitting. Variations in KE loss are less profound and generally decrease with the epochs.

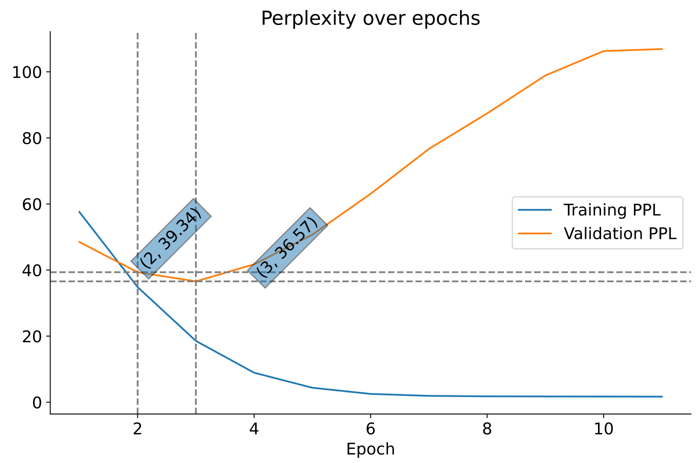

Figure 4.8: Training VS Validation perplexity for the single-labeled dataset.

The training vs validation perplexity graph (Figure 4.8) shows that the model's prediction effectiveness goes down quickly after the 2nd epoch, as can be seen by the increasing validation perplexity line. The low training perplexity indicates that the model overfits after the first few epochs.

4.3.2 Single-labeled - enriched data

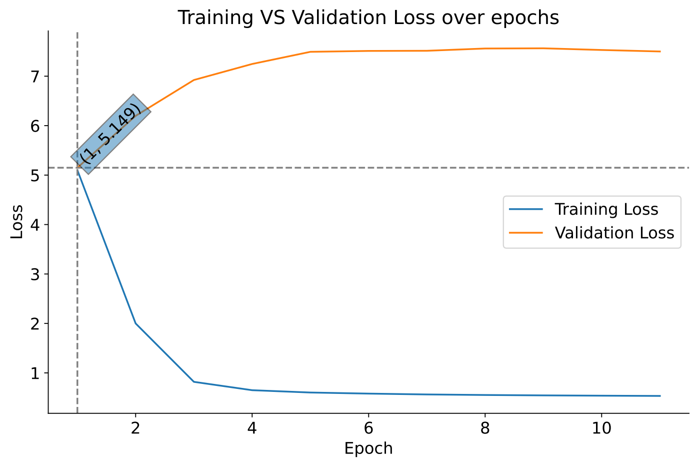

Figure 4.9: Training VS Validation loss for the single-labeled enriched dataset.

In the training vs validation loss graph (Figure 4.9), the model seems to overfit from the very beginning, since the training and the validation losses diverge, before both flattening out after around 5 epochs.

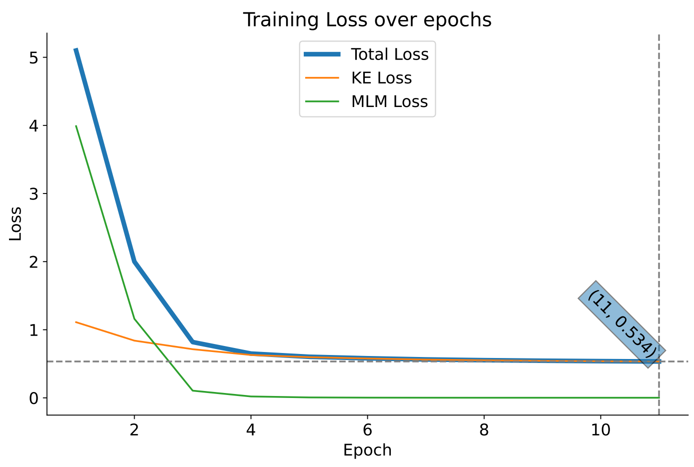

Figure 4.10: Training loss for the single-labeled enriched dataset.

It is observed in the training loss graph (Figure 4.10), that the loss values decrease within the first 5 epochs, before flattening out.

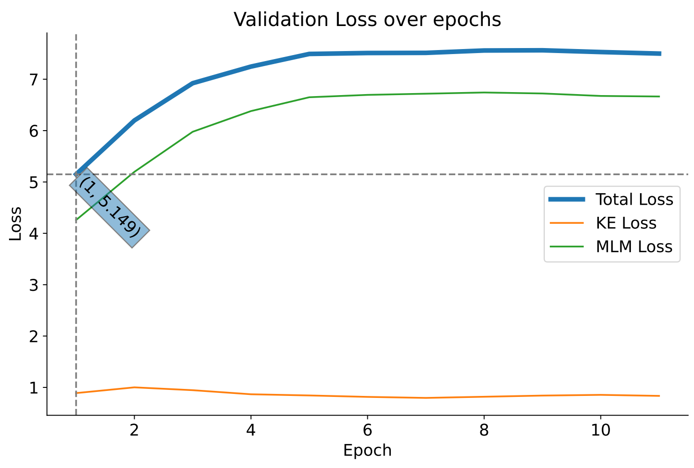

Figure 4.11: Validation loss for single-labeled enriched dataset.

In the validation loss graph (Figure 4.11), the total loss is seen to be increasing from the beginning. The KE loss remains somewhat flat, but the MLM loss increases for the first 5 epochs before flattening out.

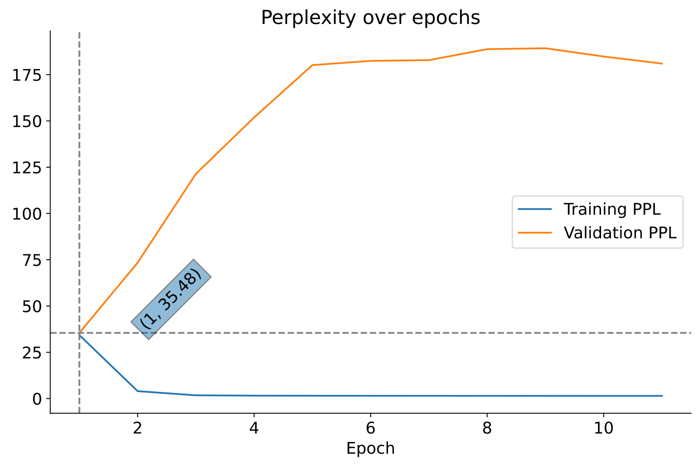

Figure 4.12: Training VS Validation perplexity for the single-labeled enriched dataset.

In line with the loss graphs, in the perplexity graph (Figure 4.12), the validation perplexity increases, showcasing the decrease in model effectiveness from the beginning. The low training perplexity indicates overfitting.

4.3.3 Multi-labeled data

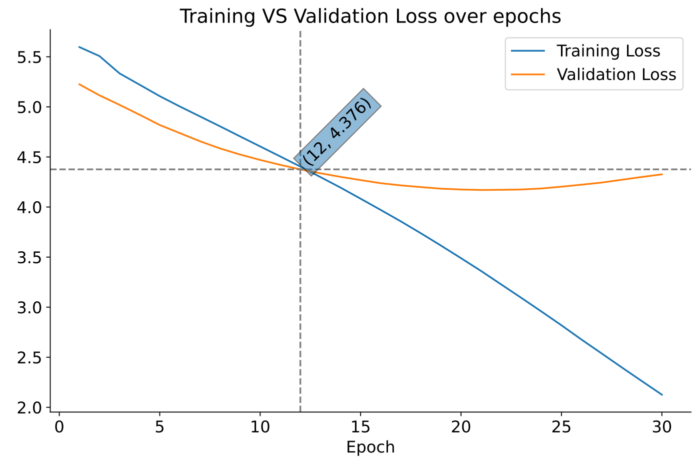

Figure 4.13: Training VS Validation loss for the multi-labeled dataset.

The training vs validation graph for multi-labeled data (Figure 4.13) shows that the model starts overfitting around the 12th epoch. The loss decrease is also more gradual than those of the models trained on single-labeled datasets (Figure 4.5, Figure 4.9), showing that this model takes more iteration to train than the single-labeled counterparts.

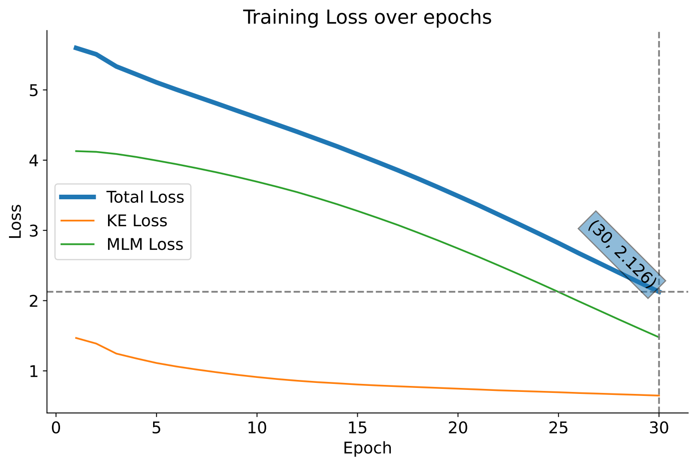

Figure 4.14: Training loss for the multi-labeled dataset.

It is seen in the training loss graph for the model trained on the multi-labeled dataset (Figure 4.14) that the loss decreases more steadily, in an almost linear fashion than the models trained on single-labeled datasets (Figure 4.6, Figure 4.10), suggesting that this model trains at a relatively slower and steadier pace.

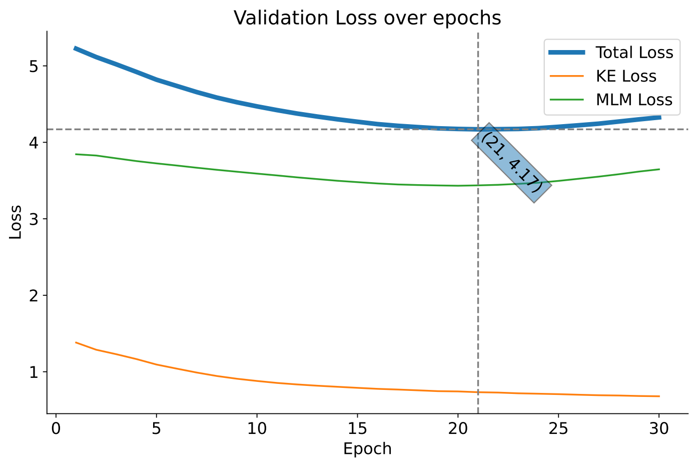

Figure 4.15: Validation loss for the multi-labeled dataset.

It is observed in the validation loss graph for the model trained on the multi-labeled dataset (Figure 4.15) that the loss gradually decreases until around the 21st epoch before starting to increase.

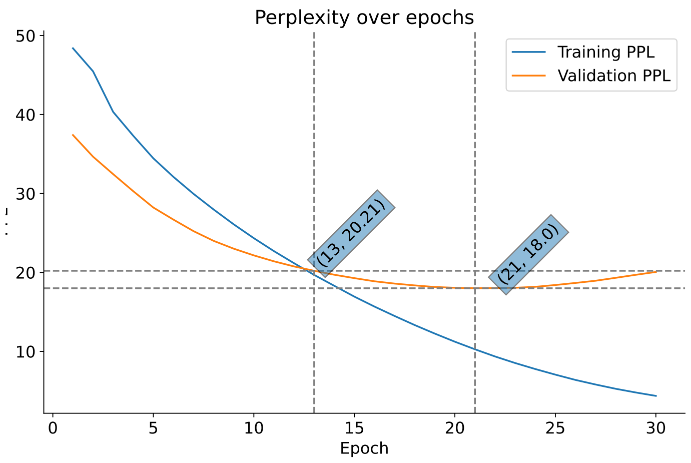

Figure 4.16: Training VS Validation perplexity for the multi-labeled dataset.

The perplexity graph (Figure 4.16) shows similar information as the loss graphs. The model starts to lose its prediction effectiveness around the 13th epoch. The lowest Validation perplexity is around the 21st epoch.

4.3.4 Multi-labeled - enriched data

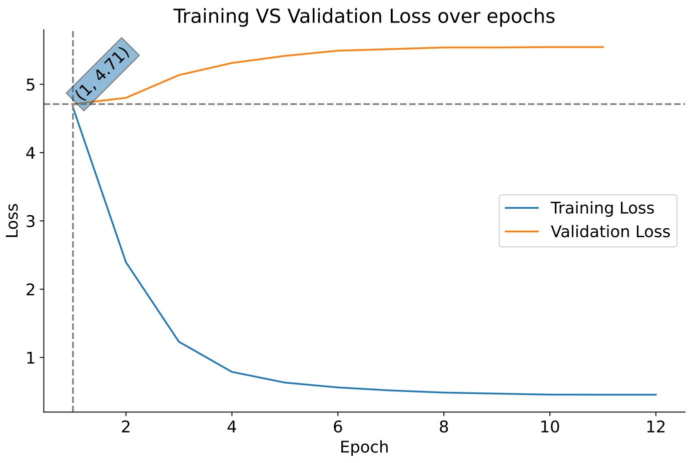

Figure 4.17: Training VS Validation loss for the multi-labeled enriched dataset.

As seen from the training vs validation loss graph for the model trained on multi-labeled enriched data (Figure 4.17), the losses diverge from the start. As such, the model checkpoint, after just one training iteration, was used for further work.

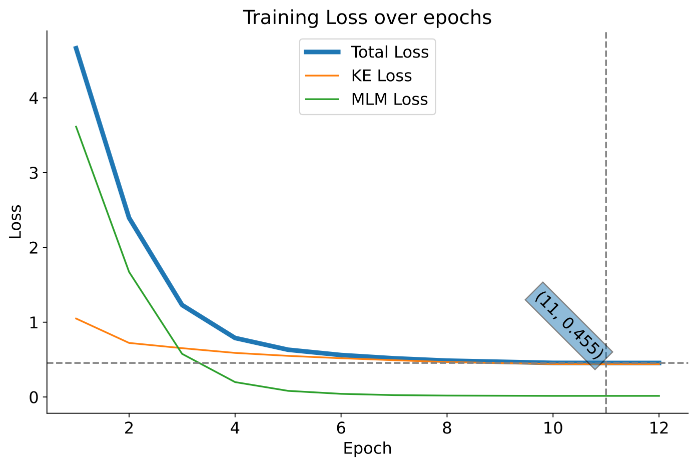

Figure 4.18: Training loss for the multi-labeled enriched dataset.

The training loss graph for the model trained on the multi-labeled dataset with enriched data (Figure 4.18) decreases quicker than the model that was trained on the multi-labeled base data (Figure 4.14). This phenomenon is similar to what was seen for models trained on single-labeled data. (Figure 4.6 vs Figure 4.10)

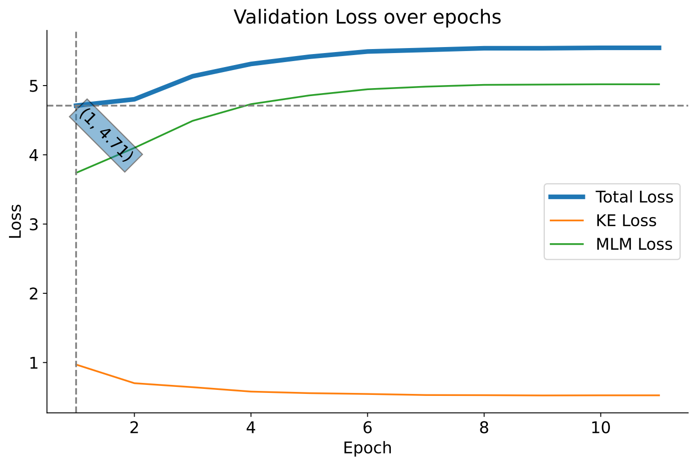

Figure 4.19: Validation loss for the multi-labeled enriched dataset.

The validation loss graph for the model trained on the multi-labeled enriched dataset (Figure 4.19) shows that the validation loss only increases with the epochs. The KE loss decreases by a little margin, but the MLM loss increase negates that improvement, indicating there is little to be gained in further training of the model.

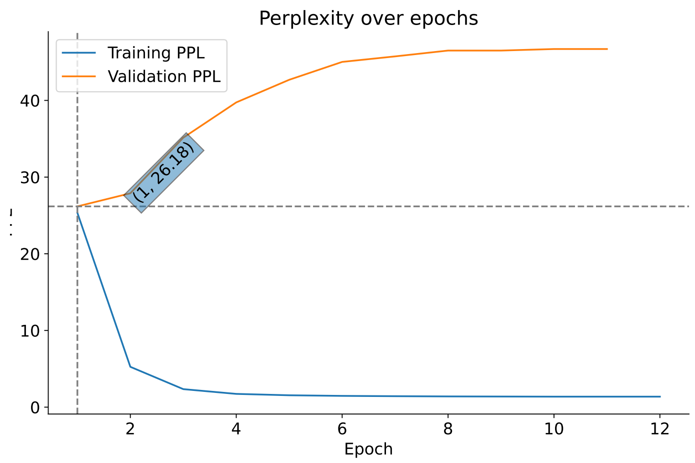

Figure 4.20: Training VS Validation perplexity for the multi-labeled enriched dataset.

In line with the loss graph (Figure 4.17), it is seen in the perplexity graph (Figure 4.20) that the validation perplexity increases with the number of epochs, indicating a reduction of prediction effectiveness.

4.4 Fine-tuning with TRAM

Below are the loss graphs of the models being fine-tuned with labeled data from TRAM, before they are evaluated, whose results are in the next section 4.5.

It is seen in general that the models overfit after 2 to 3 epochs of fine-tuning.

4.4.1 Single-labeled data

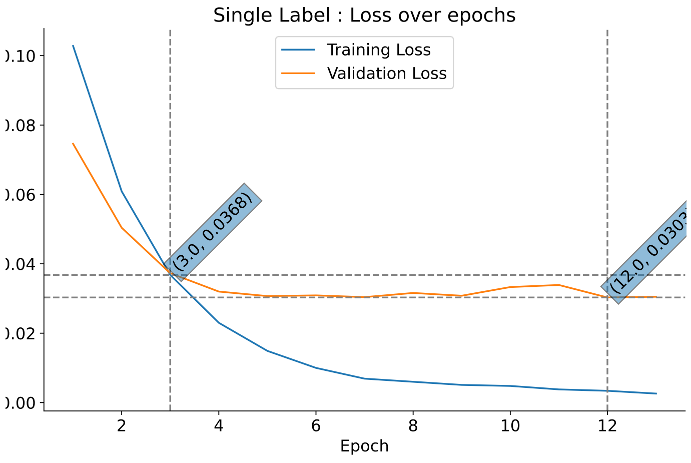

Figure 4.21: Model fine-tuned on the single-labeled data.

It is seen in the training vs validation loss graph for the model trained on single-labeled data (Figure 4.21) that the losses decrease until the 3rd epoch, before the validation loss flattens out, after which the model starts to overfit.

4.4.2 Single-labeled - enriched data

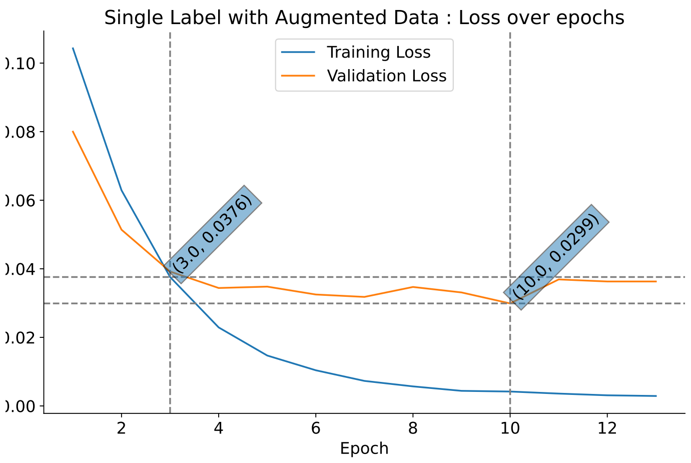

Figure 4.22: Model fine-tuned on single-labeled data along with enriched information.

The training vs validation loss graph for the model trained on single-labeled enriched data (Figure 4.22) is similar to the graph of the single-labeled base data (Figure 4.21), and the losses decrease until the 3rd epoch, after which training loss still goes down as the validation loss flattens, showing signs of overfitting.

4.4.3 Multi-labeled data

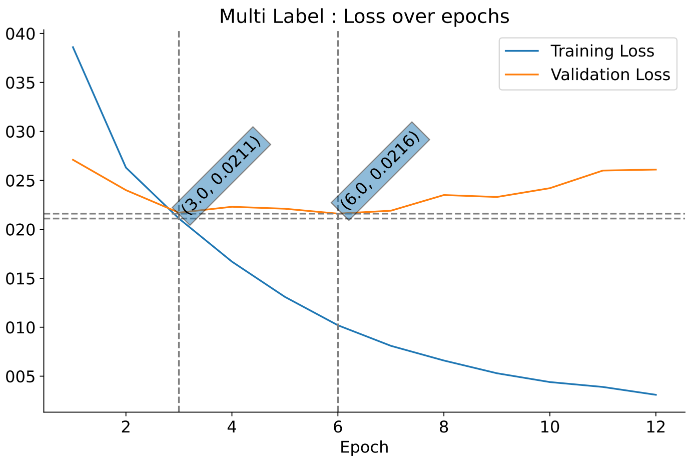

Figure 4.23: Model fine-tuned on the multi-labeled data.

The training loss decrease is smoother as seen in the loss graph for the model trained on multi-labeled data (Figure 4.23) than on the graph for the single-labeled data (Figure 4.21), which is a similar phenomenon that was observed during the KEPLER training process of multi-labeled data (Figure 4.5 vs Figure 4.13). The validation loss decreases until the 3rd epoch before flattening out.

4.4.4 Multi-labeled - enriched data

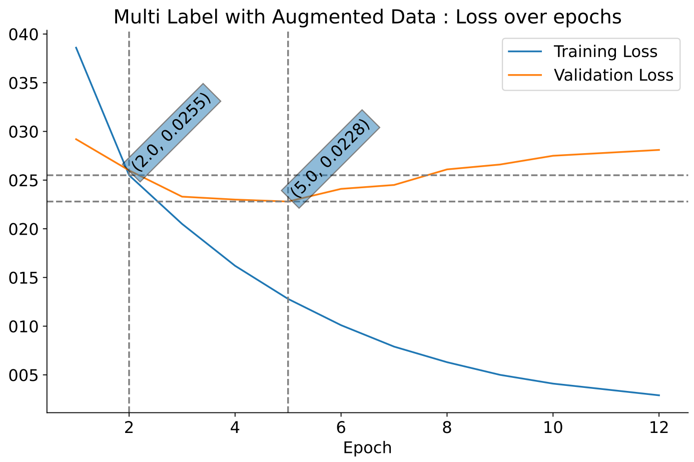

Figure 4.24: The model fine-tuned on multi-labeled data along with enriched information.

The loss graph for the model trained on multi-labeled enriched data (Figure 4.24) looks similar to the loss graph for the multi-labeled base data (Figure 4.23). However, here, the validation and the training losses meet quickly at the 2nd epoch.

4.5 Evaluation with TRAM

This section records the evaluation scores and prediction outcomes with four of the transformed KEPLER models and compares them against the performance of TRAM and a dummy untrained model. The fine-tuned KEPLER model weights, from the epochs at the point of overfitting, have been chosen for evaluation in all the below cases. The metrics used here are defined in the Theory section (2.1)

4.5.1 KEPLER model - single-labeled data

4.5.1.1 Model scores

Techniques	P	R	F1	#	Techniques	P	R	F1	#
T1003.001	0.955	1.000	0.977	21	T1105	0.750	0.983	0.851	58
T1005	0.611	0.688	0.647	16	T1106	0.700	0.946	0.805	37
T1016	0.857	0.545	0.667	11	T1110	0.833	0.909	0.870	11
T1021.001	0.722	0.963	0.825	27	T1112	0.552	0.941	0.696	17
T1027	0.924	0.841	0.881	145	T1113	1.000	0.923	0.960	13
T1033	1.000	0.273	0.429	11	T1140	0.925	0.966	0.945	89
T1036.005	0.875	0.636	0.737	11	T1190	0.786	0.917	0.846	12
T1041	0.857	0.500	0.632	12	T1204.002	0.875	0.824	0.848	17
T1047	0.786	0.917	0.846	12	T1218.011	1.000	1.000	1.000	9
T1053.005	1.000	1.000	1.000	23	T1219	1.000	0.231	0.375	13
T1055	0.952	0.967	0.959	61	T1543.003	1.000	1.000	1.000	4
T1056.001	0.636	0.875	0.737	8	T1547.001	0.667	0.714	0.690	14
T1057	0.929	0.867	0.897	15	T1548.002	1.000	0.571	0.727	7
T1059.003	0.788	1.000	0.881	63	T1562.001	0.818	0.818	0.818	11
T1070.004	1.000	0.632	0.774	19	T1566.001	1.000	0.900	0.947	20
T1071.001	0.784	0.879	0.829	33	T1570	0.632	0.857	0.727	14
T1078	0.778	0.840	0.808	25	T1573.001	1.000	0.444	0.615	9
T1082	0.829	0.850	0.840	40	T1574.002	1.000	1.000	1.000	18
T1083	0.467	0.824	0.596	17	(micro)	0.829	0.829	0.829	
T1090	0.875	0.840	0.857	25	(macro)	0.677	0.636	0.633	
T1095	1.000	0.300	0.462	10					

Table 4.5: Prediction scores for the model trained on single-labeled data.

All the micro-average scores of the KEPLER model trained on single-labeled data, as shown in the scores table (4.5), **surpasses** the TRAM model scores as shown in its prediction score table (4.13) when evaluated on the exact same test dataset.

4.5.1.2 Model predictions

#	Text excerpts	Label(s)
1	highly obfuscated and under-development custom loaders to infect those involved in the cryptocurrency	{'T1027'}
3	eyes of the victim and draw attention away from the malicious binary. Figure 3. A machine translation of	{'T1105'}
5	stage of Enigma, Interview conditions.word.exe, is a downloader written in C . Its primary	{'T1105'}
7	C . Its primary objective is to download, deobfuscate, decompress, and launch the secondary stage payload. The malware incorporates	{'T1140'}
9	examine how the malware decrypts strings and resolves hashed Windows APIs. By understanding	{'T1140'}
...	...	...
64	version) Finally, the EnigmaDownloader_s002 downloads and executes the next- stage payload on the infected system. To achieve this	{'T1105'}
66	to retrieve the command. Upon receiving the runassembly command, the malware downloads the	{'T1059.003'}
68	malware downloads the next part of the	{'T1105'}
70	chaining (CBC) mode. Figure 43. String encryption logic List of decrypted strings: \ Chromium\ User Data\ \ Google\ Chrome\ User Data\ \ Google(x86)\ Chrome\ User	{'T1140'}
72	these actors can target individuals and organizations across the cryptocurrency and Web 3 sphere. Furthermore, this case highlights the evolving nature of modular	{'T1105'}

Table 4.6: Example Technique predictions from text excerpts of a CTI report, with a prediction confidence of 0.9

The KEPLER model trained on the single-labeled dataset, predicted **23** phrases linked with some techniques, with prediction confidence of **0.9**, parts of which are shown in the predictions table (Table 4.6) above.

4.5.2 KEPLER model - single-labeled enriched data

4.5.2.1 Model scores

Techniques	P	R	F1	#
T1003.001	1.000	0.952	0.976	21
T1005	1.000	0.312	0.476	16
T1016	1.000	0.636	0.778	11
T1021.001	1.000	0.926	0.962	27
T1027	0.872	0.890	0.881	145
T1033	0.727	0.727	0.727	11
T1036.005	1.000	0.273	0.429	11
T1041	0.571	1.000	0.727	12
T1047	0.900	0.750	0.818	12
T1053.005	0.786	0.957	0.863	23
T1055	0.851	0.934	0.891	61
T1056.001	0.889	1.000	0.941	8
T1057	0.929	0.867	0.897	15
T1059.003	0.900	1.000	0.947	63
T1070.004	0.895	0.895	0.895	19
T1071.001	0.969	0.939	0.954	33
T1074.001	0.500	0.167	0.250	6
T1078	0.880	0.880	0.880	25
T1082	0.829	0.850	0.840	40
T1083	0.562	0.529	0.545	17
T1090	0.889	0.960	0.923	25
T1095	1.000	0.800	0.889	10
T1105	0.915	0.931	0.923	58
T1106	0.739	0.919	0.819	37
T1110	0.533	0.727	0.615	11
T1112	0.810	1.000	0.895	17
T1113	0.722	1.000	0.839	13
T1140	0.955	0.955	0.955	89
T1190	0.714	0.833	0.769	12
T1204.002	0.593	0.941	0.727	17
T1218.011	0.750	1.000	0.857	9
T1219	1.000	0.692	0.818	13
T1484.001	1.000	1.000	1.000	5
T1543.003	0.667	0.500	0.571	4
T1547.001	0.923	0.857	0.889	14
T1548.002	1.000	0.857	0.923	7
T1562.001	0.667	0.909	0.769	11
T1566.001	0.708	0.850	0.773	20
T1570	0.900	0.643	0.750	14
T1573.001	1.000	0.556	0.714	9
T1574.002	0.947	1.000	0.973	18
(micro)	0.851	0.851	0.851	
(macro)	0.704	0.682	0.675	

Table 4.7: Prediction scores for the model trained on single-labeled enriched data.

All the micro-average scores for the model trained on single-labeled enriched data (Table 4.7) **surpasses** the TRAM model scores (Table 4.13), and are the **highest** amongst the models that were evaluated with single-labeled data.

4.5.2.2 Model predictions

#	Text excerpts	Label(s)
1	In this campaign, the suspected Russian threat actors use several highly obfuscated and	{'T1027'}
3	highly obfuscated and under-development custom loaders to infect those involved in the cryptocurrency	{'T1027'}
5	distributed to victims via phishing attempts or through social explains how to amend	{'T1566.001'}
7	in its operation. The	{'T1055'}
9	Jobs Trend Micro (NO) MD5 1693D0A858B8FF3B83852C185880E459 SHA-1 5F1536F573D9BFEF21A4E15273B5A9852D3D81F1 SHA-03B9D7296B01E8F3FB3D12C4D80FE8A1BB0AB2FD76F33C5C E11B40729B75FB23 256 File	{'T1055'}
...	...	...
104	this case highlights the evolving nature of modular malware that employ highly obfuscated and evasive techniques along with the utilization of continuous integration	{'T1027'}
106	or phishing attempts that o	{'T1566.001'}
108	Trend Micro (NO) CONTACT US SUBSCRIBE Related Articles Trend Micro Collaborated with	{'T1027'}
110	Micro Collaborated with Interpol in Cracking Down Grandoreiro Banking Trojan NCSC Says Newer	{'T1027'}
112	personalizatRioens, soouciracl mesedia functionality and advertising. Our Cookie Notice provides more information and	{'T1027'}

Table 4.8: Example Technique predictions from text excerpts of a CTI report, with a prediction confidence of 0.9

The KEPLER model trained on the multi-labeled dataset, predicted **57** phrases linked with some techniques, with a prediction confidence of **0.9**, parts of which are shown above (Table 4.8).

4.5.3 KEPLER model - multi-labeled data

4.5.3.1 Model scores

Techniques	P	R	F1	#
T1140	0.907	0.764	0.829	89
T1055	0.766	0.692	0.727	52
T1059.003	0.927	0.535	0.679	71
T1027	0.785	0.480	0.596	152
T1105	0.909	0.208	0.339	48
T1003.001	1.000	0.160	0.276	25
T1047	1.000	0.095	0.174	21
T1106	0.500	0.098	0.163	41
T1078	1.000	0.059	0.111	34
(micro)	0.837	0.230	0.361	
(macro)	0.156	0.062	0.078	

Table 4.9: Prediction scores for the model trained on multi-labeled data.

As seen in the scores table for the model trained on multi-labeled data (Table 4.9), the micro-average Precision score is higher than the Recall and F1 Scores. It is also the **highest** amongst all models evaluated on multi-labeled data.

4.5.3.2 Model predictions

#	Text excerpts	Label(s)
1	the speci	{'T1140'}
3	The techniques used to decrypt strings and resolve API hashes in both the	{'T1140'}
5	downloading, deobfuscating, decompressing, and renaming the downloaded payload This website uses cookies	{'T1140'}
7	21. Payload deobfuscation and decompression Before executing the payload, the malware attempts to	{'T1140'}

Table 4.10: Example Technique predictions from text excerpts of a CTI report, with a prediction confidence of 0.9.

The KEPLER model trained on the multi-labeled dataset, predicted **4** sentences linked to the same technique T1440 with a prediction confidence of **0.9**, as seen in the prediction table (4.10).

4.5.4 KEPLER model - multi-labeled enriched data

4.5.4.1 Model scores

Techniques	P	R	F1	#
T1140	0.819	0.764	0.791	89
T1027	0.552	0.559	0.556	152
T1059.003	0.871	0.380	0.529	71
T1055	1.000	0.038	0.074	52
(micro)	0.674	0.177	0.280	
(macro)	0.065	0.035	0.039	

Table 4.11: Prediction scores for the model trained on multi-labeled enriched data.

As can be observed from the scores table for the model trained on multi-labeled enriched data (Table 4.11), the micro and macro average scores were the lowest among the four KEPLER models, as well as reference models that were evaluated.

4.5.4.2 Model predictions

#	Text excerpts	Label(s)
1	`to download, deobfuscate, decompress, and launch the secondary stage payload. The malware incorporates`	{'T1140'}
3	`The techniques used to decrypt strings and resolve API hashes in both the`	{'T1140'}
5	`Figure 18. The code responsible for decrypting the next stage payload`	{'T1140'}
7	`demonstrates how the malware downloads, deobfuscates, and decompresses`	{'T1140'}
9	`services URLs, are encrypted with the AES algorithm in cipher-block chaining (CBC) mode.`	{'T1027'}

Table 4.12: Example Technique predictions from text excerpts of a CTI report, with a prediction confidence of 0.9.

The KEPLER model, trained on the multi-labeled enriched dataset (Table 4.12), predicted **5** sentences linked with two techniques, with a confidence of **0.9**.

4.5.5 Reference Models

Here, we showcase the predictive performance of the TRAM models, trained on and evaluated with both single and multi-labeled data, and a dummy untrained model.[75]

[75] https://huggingface.co/hf-internal-testing/tiny-random-BertModel

4.5.5.1 TRAM - single-labeled

Techniques	P	R	F1	#
T1003.001	0.857	0.571	0.686	21
T1005	0.750	0.375	0.500	16
T1012	0.714	1.000	0.833	5
T1016	0.474	0.818	0.600	11
T1021.001	0.880	0.815	0.846	27
T1027	0.728	0.814	0.769	145
T1033	0.833	0.909	0.870	11
T1036.005	0.280	0.636	0.389	11
T1041	1.000	0.417	0.588	12
T1047	0.917	0.917	0.917	12
T1053.005	0.958	1.000	0.979	23
T1055	0.939	0.754	0.836	61
T1056.001	0.727	1.000	0.842	8
T1057	0.824	0.933	0.875	15
T1059.003	0.959	0.746	0.839	63
T1068	0.222	0.667	0.333	3
T1070.004	0.739	0.895	0.810	19
T1071.001	0.862	0.758	0.806	33
T1074.001	0.273	0.500	0.353	6
T1078	0.818	0.720	0.766	25
T1082	0.842	0.800	0.821	40
T1083	0.667	0.941	0.780	17
T1090	0.500	0.920	0.648	25
T1095	0.636	0.700	0.667	10
T1105	0.759	0.759	0.759	58
T1106	0.821	0.622	0.708	37
T1110	0.667	0.909	0.769	11
T1112	0.889	0.941	0.914	17
T1113	1.000	0.846	0.917	13
T1140	0.915	0.843	0.877	89
T1190	0.833	0.417	0.556	12
T1204.002	0.857	0.706	0.774	17
T1210	0.333	0.250	0.286	4
T1218.011	0.818	1.000	0.900	9
T1219	0.417	0.385	0.400	13
T1518.001	0.500	1.000	0.667	3
T1543.003	0.444	1.000	0.615	4
T1547.001	0.846	0.786	0.815	14
T1548.002	0.600	0.857	0.706	7
T1552.001	0.500	0.200	0.286	5
T1562.001	0.643	0.818	0.720	11
T1564.001	0.500	1.000	0.667	3
T1566.001	0.889	0.800	0.842	20
T1569.002	0.286	0.667	0.400	3
T1570	1.000	0.143	0.250	14
T1573.001	1.000	0.111	0.200	9
T1574.002	0.850	0.944	0.895	18
(micro)	0.758	0.758	0.758	
(macro)	0.689	0.706	0.659	

Table 4.13: Prediction scores for the TRAM model trained on single-labeled data.

The prediction scores of TRAM on single-labeled data (Table 4.13) is the reference against which our KEPLER models are compared against.

4.5.5.2 TRAM - multi-labeled

Techniques	P	R	F1	#
T1574.002	1.000	0.917	0.957	12
T1021.001	0.944	0.850	0.895	20
T1140	0.820	0.820	0.820	89
T1047	1.000	0.667	0.800	21
T1059.003	0.891	0.690	0.778	71
T1105	0.755	0.771	0.763	48
T1562.001	1.000	0.609	0.757	23
T1056.001	1.000	0.600	0.750	5
T1090	0.875	0.636	0.737	22
T1027	0.803	0.645	0.715	152
T1055	0.861	0.596	0.705	52
T1053.005	0.875	0.560	0.683	25
T1003.001	0.875	0.560	0.683	25
T1070.004	0.917	0.524	0.667	21
T1071.001	0.789	0.577	0.667	26
T1106	0.667	0.585	0.623	41
T1218.011	1.000	0.400	0.571	15
T1204.002	1.000	0.357	0.526	28
T1566.001	0.800	0.267	0.400	15
T1112	0.545	0.300	0.387	20
T1083	1.000	0.238	0.385	21
T1041	0.364	0.400	0.381	10
T1078	0.889	0.235	0.372	34
T1082	0.800	0.154	0.258	26
T1548.002	1.000	0.143	0.250	7
T1113	1.000	0.111	0.200	9
T1570	0.500	0.091	0.154	11
(micro)	0.825	0.474	0.602	
(macro)	0.459	0.266	0.318	

Table 4.14: Prediction scores for the TRAM model trained on multi-labeled data.

As noted in the TRAM documentation[76], the scores for this model, which has been trained on multi-labeled data (Table 4.14), are lower than the TRAM's single-labeled model scores (Table 4.13).

4.5.5.3 Dummy model - single-labeled

An evaluation was run on an untrained dummy model (`tiny-random-BertModel`)[77] as a control.

Techniques	P	R	F1	#
T1090	0.025	1.000	0.048	25
(micro)	0.025	0.025	0.025	
(macro)	0.001	0.020	0.001	

Table 4.15: Prediction scores for the dummy model trained on single-labeled data.

[76] https://github.com/center-for-threat-informed-defense/tram/wiki/Data-Annotation#single-label-vs-multi-label

[77] See footnote 75

From the prediction score table for the dummy model trained on single-labeled data (Table 4.15), it can be observed that it has the lowest scores amongst all models trained on single-labeled data.

4.5.5.4 Dummy model - multi-labeled

Techniques	P	R	F1	#
T1055	0.014	1.000	0.027	52
T1105	0.013	1.000	0.025	48
T1078	0.009	1.000	0.018	34
T1204.002	0.007	1.000	0.014	28
T1071.001	0.007	1.000	0.013	26
T1082	0.007	1.000	0.013	26
T1562.001	0.006	1.000	0.012	23
T1047	0.005	1.000	0.011	21
T1547.001	0.005	1.000	0.010	20
T1566.001	0.004	1.000	0.008	15
T1543.003	0.004	1.000	0.007	14
T1218.011	0.003	0.400	0.006	15
T1574.002	0.003	1.000	0.006	12
T1005	0.003	1.000	0.006	12
T1570	0.003	1.000	0.006	11
T1095	0.003	1.000	0.006	11
T1016	0.002	1.000	0.005	9
T1113	0.002	1.000	0.005	9
T1110	0.002	1.000	0.004	8
T1518.001	0.002	1.000	0.004	7
T1033	0.002	1.000	0.004	7
T1190	0.002	1.000	0.003	6
T1484.001	0.002	1.000	0.003	6
T1552.001	0.002	1.000	0.003	6
T1056.001	0.001	1.000	0.003	5
T1210	0.001	1.000	0.003	5
T1012	0.001	1.000	0.003	5
T1569.002	0.001	1.000	0.002	4
T1068	0.001	1.000	0.002	3
(micro)	0.004	0.426	0.008	
(macro)	0.002	0.568	0.005	

Table 4.16: Prediction scores for the dummy model trained on multi-labeled data.

From the Prediction score table for the dummy model trained on multi-labeled data (Table 4.16), it can be observed that other than the macro Recall score, it has the lowest values amongst all models trained on multi-labeled data.

4.5.6 Comparisons

4.5.6.1 Precision, Recall & F1 scores

Here we consolidate the micro and macro average scores, for easier comparison. The best scores are highlighted in green, whereas the second-highest scores are highlighted in yellow.

Micro average	P	R	F1
KEPLER Single Labelled Data	0.829	0.829	0.829
KEPLER Single Labelled Enriched Data	0.851	0.851	0.851
TRAM Single Labelled Data	0.758	0.758	0.758
Dummy Single Labelled Data	0.025	0.025	0.025

Table 4.17: Micro average scores for models trained on single-labeled data.

As seen in the table that compares the micro average scores for models trained on single-labeled data (Table 4.17), KEPLER model trained on single-labeled enriched data is seen to have the **highest micro average scores**, followed by the KEPLER model trained on the single-labeled base data.

Macro average	P	R	F1
KEPLER Single Labelled Data	0.677	0.636	0.633
KEPLER Single Labelled Enriched Data	0.704	0.682	0.675
TRAM Single Labelled Data	0.689	0.706	0.659
Dummy Single Labelled Data	0.001	0.020	0.001

Table 4.18: Macro average scores for models trained on single-labeled data.

In the table that compares macro average scores for models trained on single-labeled data (Table 4.18), it is seen that the KEPLER model trained on single-labeled enriched data has **higher** macro average Precision and F1 Scores, while the TRAM model trained on single-labeled data has the better macro Recall score.

Micro average	P	R	F1
KEPLER Multi Labelled Data	0.837	0.230	0.361
KEPLER Multi Labelled Enriched Data	0.674	0.177	0.280
TRAM Multi Labelled Data	0.825	0.474	0.602
Dummy Multi Labelled Data	0.004	0.426	0.008

Table 4.19: Micro average scores for models trained on multi-labeled data.

The table, comparing the micro average scores for models trained on multi-labeled data (Table 4.19), generally shows lower performance than the models trained on single-labeled data (Table 4.17). But, the KEPLER model trained on multi-labeled data has the **highest** micro Precision scores amongst the other competing models. TRAM model trained on multi-labeled data has better Recall and F1 Scores.

Macro average	P	R	F1
KEPLER Multi Labelled Data	0.156	0.062	0.078
KEPLER Multi Labelled Enriched Data	0.065	0.035	0.039
TRAM Multi Labelled Data	0.459	0.266	0.318
Dummy Multi Labelled Data	0.002	0.568	0.005

Table 4.20: Macro average scores for models trained on multi-labeled data.

The table that compares macro average scores for models trained on multi-labeled data (Table 4.20) also shows lower performance than the models trained on single-labeled data (Table 4.18). TRAM model trained on multi-labeled data has better macro Precision and F1 Scores. The dummy model here has the highest macro Recall score.

4.5.6.2 Actual vs predicted technique counts

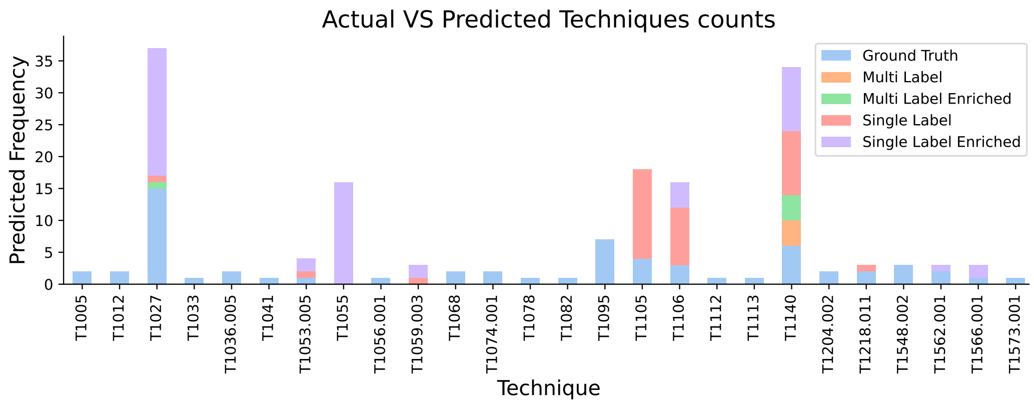

Figure 4.25: This figure compares the actual and predicted technique counts in the CTI report named ***`Enigma Stealer Targets Cryptocurrency Industry with Fake Jobs`***

In the stacked bar plot above (Figure 4.25), the x-axis represents a combined list of techniques predicted by the four models under test, along with the annotated techniques from the TRAM input data, which we treat as the ground truth. The y-axis shows the number of times a technique is predicted or annotated. All of this is in respect to one CTI report named `Enigma Stealer Targets Cryptocurrency Industry with Fake Jobs`.

It can be seen above, that the best performing model (KEPLER single-labeled with Enriched Data), in purple, matches closely in technique prediction counts with the ground truth, in blue, followed by the second best performing model (KEPLER single-labeled Data) in peach, especially with the count of the overrepresented techniques like `T1027` and `T1140`, which also explains why the micro-average scores were better than the macro-average scores for these models.

4.5.6.3 Validation losses while training with KEPLER

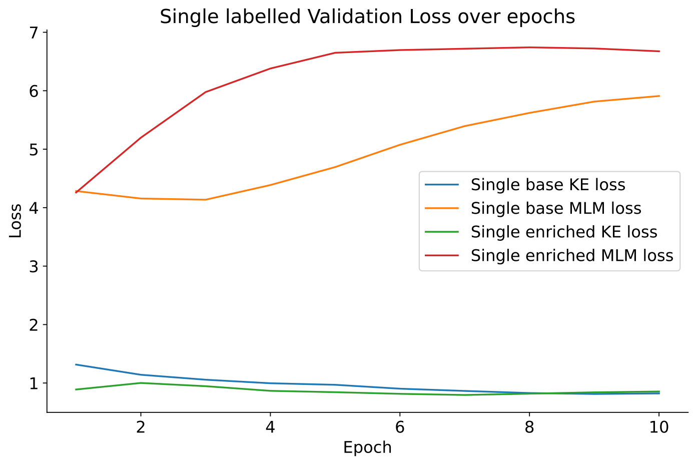

Figure 4.26: This figure compares the MLM and KE losses for the models trained on ***single*** *labeled base and enhanced datasets.*

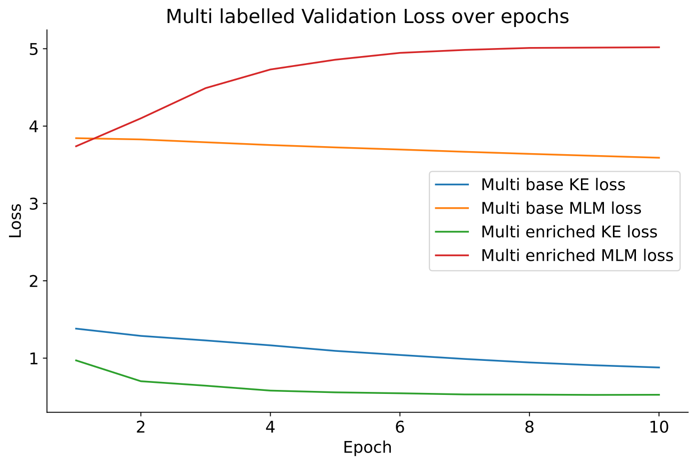

Figure 4.27: This figure compares the MLM and KE losses for the models trained on ***multi*** *labeled base and enhanced datasets.*

It is observed that the KE validation losses are lower for the models trained on enriched datasets than the ones trained on the base datasets. And vice versa for the MLM losses, where they are higher for the models trained on the enhanced datasets. (Figure 4.26, Figure 4.27)

Chapter 5

Discussion

The objective of this study was to explore the efficacy of Knowledge Graphs in enhancing the prediction capability of models dealing with classification of Cyber Threat Intelligence report text excerpts to corresponding MITRE ATT&CK techniques. This chapter discusses the findings in relation to existing literature, evaluates the strengths and weaknesses of the study, and considers the practical implications and future research avenues.

5.1 Models trained on single-labeled data

As an outcome of this study, it was found that the KEPLER based model trained on the single-labeled enriched dataset yielded the highest micro-average P, R, F1 scores, and macro-average P, F1 scores(Table 4.17, Table 4.18), where the model's ability to learn from semantic relationships through its KE objective was demonstrated, in contrast to the TRAM model trained on the single-labeled dataset, which solely relies on Language Model objectives. This was possibly the biggest distinguishing factor between the two models. The enrichment of additional data from the MITRE ATT&CK dataset seems to have further positively influenced the model's performance, specifically the Knowledge Embedding objective (Figure 4.26). The fact that the quality of annotations in the single-labeled TRAM data is superior to that of the multi-labeled data[78], also added up to the performance improvement.

[78]https://github.com/center-for-threat-informed-defense/tram/wiki/Data-Annotation#single-label-vs-multi-label

The second-highest micro-average P, R, F1 scores were seen for the KEPLER based model trained on the single-labeled base dataset, possibly due to the absence of the data enrichment that was done for the highest-performing model. However, the macro prediction scores were not as high, likely due to the imbalance in the dataset (Figure 4.1, Tables 4.17 and 4.18).

The models trained on single-labeled data were found to be equally proficient at correctly predicting positive instances and avoiding false positives despite data imbalance, demonstrating consistent performance across most instances and classes. As observed by the similar micro and macro average P and R scores of each corresponding model (Table 4.17, Table 4.18).

Both our single-label trained models, which were pre-trained on the broad WikiData5m[79], despite not being fine-tuned for predicting Attack Techniques from CTI reports, surpassed the TRAM initiative's performance in micro and macro average prediction scores. TRAM, which utilizes a newer language model pre-trained on a more specialized SciBERT[43] dataset, tends to underperform when exposed to larger datasets, leading to their focus on the top 50 techniques[80]. On the other hand, our models demonstrated better predictive capabilities upon integrating additional relational data, suggesting greater robustness and scalability. (Table 4.17, Table 4.18). This is reflected in the training KE validation loss graph (Figure 4.26), which show lower KE losses for the model trained on the enriched dataset compared to the base dataset, indicating that the inclusion of additional relational data beneficially influences KE predictions. However, the MLM validation losses suggest that an increase in data volume adversely affects predictions. This reinforces our stance that using a KE model with relational data along with an LLM is a strategic move for effective predictions over larger datasets.

5.2 Models trained on multi-labeled data

For the KEPLER based model trained on the multi-labeled base dataset, high micro Precision but low Recall was observed, indicating that the model is making fewer positive predictions (Section 4.5.3), but a higher proportion of these predictions are correct. The model is likely producing a significant

[79]https://deepgraphlearning.github.io/project/wikidata5m

[80]https://github.com/center-for-threat-informed-defense/tram/wiki/Large-Language-Models#subset-of-techniques

number of false negatives. The model's behavior suggests a preference for minimizing errors in predicting positive instances, even at the cost of missing several actual positive instances.

For all models trained on multi-labeled data, the macro average scores were found to be worse than the micro average scores. Even though the TRAM scores were relatively better, the Precision scores were significantly higher than the Recall scores. This discrepancy could be attributed to class imbalance (Figure 4.3) or model bias. The models might be biased towards predicting the majority class, resulting in a high number of false negatives and hence a lower Recall. This is because Recall is sensitive to false negatives. Additionally, there could be a significant variation in the model's performance across different classes.

In general, theKEPLER models trained on multi-labeled datasets performed much worse than their single-labeled counterparts. In contrast, the TRAM model, when trained on a multi-labeled dataset, demonstrated steadier performance for both micro and macro average scores, possibly due to its weights tuned specifically for this purpose.

The work presented in this thesis indicates that the outcomes, while somewhat positive, represent only a preliminary exploration into a significantly larger and varied area of active research, ranging from employing LLMs to analyze graph-structured information [29] and their utilization in cybersecurity [30, 10, 13], development of CKGs and their enhancement using GCNs [31], to employing ChatGPT to classify attack elements and their relationships for efficient KG construction [33], to name a few. Further work in this area is essential, to figure out the key factors that influence model performance. The impact of training the model on a significantly larger set of labeled data must be explored. Additional insights could be gained by augmenting the dataset with synthetic or negative samples. The selection of a base model pre-trained on domain-specific information, such as SciBERT[43], SecBERT[81] or SecureBERT[44], warrants consideration, in which case corresponding KGs would also need to be defined, posing questions on how a KG ontology could affect the outcome. Finally, thorough hyperparameter optimizations should be carried out for better model tuning.

[81]https://huggingface.co/jackaduma/SecBERT

To sum up, we recognize that the combination of KE and MLM objectives in model training is a powerful tool that could be used for a variety of purposes. If there are relational elements that can be extracted from a text corpus, our recommendation is to form an ontology and create a KG with that information, so that it can be used for a KE objective, jointly with LLMs. This would most likely have a positive effect on the model's predictive performance.

Chapter 6

Conclusion

This research aimed to demonstrate that the combination of KE and PLMs can measurably improve the predictive capabilities of a method that uses both techniques in conjunction, than a model that relies solely on traditional ML techniques like PLMs. The findings indicate that our two best models demonstrated higher Precision, Recall, and F1 scores in comparison to the reference TRAM model when all the models were trained and evaluated with the same data. These results support the proposed method's effectiveness in predicting Attack Techniques from CTI reports. The transformation of the TRAM dataset to a KG, and the successful application of extending the KEPLER model to work on the TRAM dataset, highlights the potential of this approach in real-world scenarios. This thesis contributes to the cyber-security domain by providing a framework that leverages the power of KGs for the automated analysis of cyber threats.

While this work has advanced our understanding of the effectiveness of knowledge graphs in predicting attack techniques from CTI reports, there remain several avenues for further investigation. Future studies could explore building more robust models that work better on imbalanced and sparsely labeled datasets. Such endeavors would not only build upon the foundation laid by this thesis but also pave the way for more resilient digital ecosystems.

Bibliography

[1] Marc Ohm and Charlene Stuke. “SoK: Practical Detection of Software Supply Chain Attacks”. In: *Proceedings of the 18th International Conference on Availability, Reliability and Security*. ACM, 2023. DOI: 10.1145/3600160.3600162.

[2] Dan Geer, Bentz Tozer, and John Speed Meyers. “For Good Measure: Counting Broken Links: A Quant’s View of Software Supply Chain Security”. In: *login Usenix Mag.* 45.4 (2020).

[3] Triet H. M. Le, Huaming Chen, and M. Ali Babar. “A Survey on Data-driven Software Vulnerability Assessment and Prioritization”. In: *ACM Computing Surveys* 55.5 (Dec. 2022), pp. 1–39. ISSN: 1557-7341. DOI: 10.1145/3529757. URL: http://dx.doi.org/10.1145/3529757.

[4] Saqib Saeed et al. “A Systematic Literature Review on Cyber Threat Intelligence for Organizational Cybersecurity Resilience”. In: *Sensors* 23.16 (2023). ISSN: 1424-8220. DOI: 10.3390/s23167273. URL: https://www.mdpi.com/1424-8220/23/16/7273.

[5] Christopher S. Johnson et al. *Guide to Cyber Threat Information Sharing*. Oct. 2016, pp. 4–5. DOI: 10.6028/nist.sp.800-150. URL: http://dx.doi.org/10.6028/NIST.SP.800-150.

[6] Matthias Fey and Jan Eric Lenssen. *Fast Graph Representation Learning with PyTorch Geometric*. 2019. arXiv: 1903.02428 [cs.LG].

[7] Dieter Fensel et al. “Introduction: What Is a Knowledge Graph?” In: Cham: Springer International Publishing, 2020, pp. 1–10.

[8] Yuke Ma et al. “The Advancement of Knowledge Graphs in Cybersecurity: A Comprehensive Overview”. In: *Computational and Experimental Simulations in Engineering*. Ed. by Shaofan Li. Cham: Springer International Publishing, 2024, pp. 65–103.

[9] Kai Liu et al. "Recent Progress of Using Knowledge Graph for Cybersecurity". In: *Electronics* 11.15 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11152287. URL: https://www.mdpi.com/2079-9292/11/15/2287.

[10] Leslie F. Sikos. "Cybersecurity knowledge graphs". In: *Knowledge and Information Systems* 65.9 (Sept. 1, 2023), pp. 3511–3531. DOI: 10.1007/s10115-023-01860-3. URL: https://doi.org/10.1007/s10115-023-01860-3.

[11] Yanai Elazar et al. "Measuring and Improving Consistency in Pretrained Language Models". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 1012–1031. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00410. URL: http://dx.doi.org/10.1162/tacl_a_00410.

[12] Xiou Ge et al. *Knowledge Graph Embedding: An Overview.* 2023. arXiv: 2309.12501 [cs.AI].

[13] Siwar Kriaa and Yahia Chaabane. "SecKG: Leveraging attack detection and prediction using knowledge graphs". In: *2021 12th International Conference on Information and Communication Systems (ICICS).* IEEE, May 2021. DOI: 10.1109/icics52457.2021.9464587. URL: http://dx.doi.org/10.1109/icics52457.2021.9464587.

[14] Xiaozhi Wang et al. "KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 176–194. DOI: 10.1162/tacl_a_00360.

[15] Aritran Piplai et al. "Creating Cybersecurity Knowledge Graphs From Malware After Action Reports". In: *IEEE Access* 8 (2020), pp. 211691–211703. DOI: 10.1109/ACCESS.2020.3039234.

[16] Injy Sarhan and Marco Spruit. "Open-CyKG: An Open Cyber Threat Intelligence Knowledge Graph". In: *Knowledge-Based Systems* 233 (2021), p. 107524. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.107524. URL: https://www.sciencedirect.com/science/article/pii/S0950705121007863.

[17] Diksha Khurana et al. "Natural language processing: state of the art, current trends and challenges". In: *Multimedia Tools and Applications* 82.3 (Jan. 1, 2023), pp. 3713–3744. DOI: 10.1007/s11042-022-13428-4. URL: https://doi.org/10.1007/s11042-022-13428-4.

[18] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. "Natural language processing: an introduction". In: *Journal of the*

American Medical Informatics Association 18.5 (Sept. 2011), pp. 544–551. ISSN: 1067-5027. DOI: 10.1136/amiajnl-2011-000464. eprint: https://academic.oup.com/jamia/article-pdf/18/5/544/5962687/18-5-544.pdf. URL: https://doi.org/10.1136/amiajnl-2011-000464.

[19] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[20] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[21] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[22] Humza Naveed et al. *A Comprehensive Overview of Large Language Models*. 2024. arXiv: 2307.06435 [cs.CL].

[23] Ed S. Ma. *Investigating Masking-based Data Generation in Language Models*. 2023. arXiv: 2307.00008 [cs.CL].

[24] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units". In: (Aug. 2015).

[25] Aidan Hogan et al. "Knowledge Graphs". In: *ACM Computing Surveys* 54.4 (2021), pp. 1–37. DOI: 10.1145/3447772.

[26] De Giacomo Giuseppe and Lenzerini Maurizio. "TBox and ABox Reasoning in Expressive Description Logics". In: *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR'96)*. 1996, pp. 37–48.

[27] Jayeeta Majumder and Saikat Khanra. "An Overview of Semantic Networks and Its Components". In: *International Journal of Engineering Research and Technology (IJERT)* (2018). URL: https://www.ijert.org/an-overview-of-semantic-networks-and-its-components.

[28] Lukas Schmelzeisen, Corina Dima, and Steffen Staab. "Wikidated 1.0: An Evolving Knowledge Graph Dataset of Wikidata's Revision History". In: *arXiv preprint arXiv:2112.05003* (2021).

[29] Bowen Jin et al. *Large Language Models on Graphs: A Comprehensive Survey*. 2024. arXiv: 2312.02783 [cs.CL].

[30] Farzad Nourmohammadzadeh Motlagh et al. *Large Language Models in Cybersecurity: State-of-the-Art*. 2024. arXiv: 2402.00891 [cs.CR].

[31] Soham Dasgupta et al. "Cybersecurity Knowledge Graph Improvement with Graph Neural Networks". In: *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021. DOI: 10.1109/bigdata52589.2021.9672062.

[32] Md Rayhanur Rahman and Laurie Williams. *From Threat Reports to Continuous Threat Intelligence: A Comparison of Attack Technique Extraction Methods from Textual Artifacts*. 2022. arXiv: 2210.02601 [cs.CR].

[33] Jiehui Liu and Jieyu Zhan. "Constructing Knowledge Graph from Cyber Threat Intelligence Using Large Language Model". In: *2023 IEEE International Conference on Big Data (BigData)*. 2023, pp. 516–521. DOI: 10.1109/BigData59044.2023.10386611.

[34] Myle Ott et al. *fairseq: A Fast, Extensible Toolkit for Sequence Modeling*. 2019. arXiv: 1904.01038 [cs.CL].

[35] Alex Wang et al. *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. 2019. arXiv: 1804.07461 [cs.CL].

[36] Guokun Lai et al. *RACE: Large-scale ReAding Comprehension Dataset From Examinations*. 2017. arXiv: 1704.04683 [cs.CL].

[37] Pranav Rajpurkar et al. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. 2016. arXiv: 1606.05250 [cs.CL].

[38] Xu Han et al. "FewRel: A Large-Scale Supervised Few-Shot Relation Classification Dataset with State-of-the-Art Evaluation". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Ed. by Ellen Riloff et al. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 4803–4809. DOI: 10.18653/v1/D18-1514. URL: https://aclanthology.org/D18-1514.

[39] Jake Snell, Kevin Swersky, and Richard S. Zemel. *Prototypical Networks for Few-shot Learning*. 2017. arXiv: 1703.05175 [cs.LG].

[40] Tianyu Gao et al. "FewRel 2.0: Towards More Challenging Few-Shot Relation Classification". In: Jan. 2019, pp. 6251–6256. DOI: 10.18653/v1/D19-1649.

[41] Livio Baldini Soares et al. *Matching the Blanks: Distributional Similarity for Relation Learning*. 2019. arXiv: 1906.03158 [cs.CL].

[42] Ruobing Xie et al. "Representation learning of knowledge graphs with entity descriptions". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 2659–2665.

[43] Iz Beltagy, Kyle Lo, and Arman Cohan. "SciBERT: Pretrained Language Model for Scientific Text". In: *EMNLP*. 2019. eprint: arXiv:1903.10676.

[44] Ehsan Aghaei et al. “SecureBERT: A Domain-Specific Language Model for Cybersecurity”. In: *Security and Privacy in Communication Networks*. Ed. by Fengjun Li et al. Cham: Springer Nature Switzerland, 2023, pp. 39–56. ISBN: 978-3-031-25538-0.