Analysing and Scoring Cybersecurity Vulnerability Reports Using Deep Learning

A thesis

submitted in partial fulfilment

of the requirements for the Degree

of

Doctor of Philosophy in Computer Science

at

The University of Waikato

by

Zijing Zhang

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2026

Abstract

Vulnerability analysis plays a crucial role in cybersecurity, as it helps identify and mitigate potential threats to systems and networks. It is essential for organizations to understand the severity of vulnerabilities and prioritize their remediation efforts. This thesis focuses on the analysis and scoring of cybersecurity vulnerability reports using deep learning techniques. By leveraging large language models and machine learning algorithms, this thesis aims to improve the accuracy and efficiency of vulnerability classification and severity assessment.

The research presented in this thesis includes the development of models to estimate the CVSS of vulnerabilities based on their descriptions, comparing the performance of AI models with human experts. For CVSS 3.1 severity level prediction task, the USE model achieves an accuracy of 0.73. The USE model outperforms human experts with a macro mean squared error of 0.128, which is four times lower than that of human experts (0.537). Meanwhile, quality of vulnerability descriptions is also analyzed to understand its impact on the performance of machine learning models. A BERT-based model ('DesQ') is proposed to assess the quality of vulnerability descriptions, achieving an accuracy of 0.76. A dataset of vulnerability descriptions with quality labels is curated and analyzed to understand the distribution of description quality across different vendors and reporting agencies. The analyzer BART model ('EVAL') identifies missing essential vulnerability aspects. Trained on a specialized quality dataset of 450 manually labeled entries, EVAL identifies Vulnerability Impact as the most frequently missing component across analyzed weaknesses. This thesis also explores the use of explainable AI techniques to provide insights into the decision-making process of machine learning models, enhancing their interpretability and trustworthiness in mapping root causes to vulnerabilities.

Acknowledgements

Other than my supervisors Dr.Pfahringer, Dr.Kumar, Dr.Mike Mayo, and Dr.Bifet, I want to thank:

My Mother Qingling for her unconditional support throughout my education.

My Former Boss Mr. Canfora for his kind adoption of my cat Nyx in the time of urgency and his pursuit of human rights and justice.

Dr. Smith for his humor, recommendation, and compassion during my first year in New Zealand.

Bhojraj for his support during my difficult time.

Dongdong for her wake-up calls of the importance of mental health.

Rea for her beautiful smiles that brighten the world.

Wairua Puoro, Tinana Utunga; Whakapono Māramatanga, Waimarie Ora

Contents

1 Introduction **1**

2 Background **6**

2.1 Machine Learning 6

2.1.1 Term Frequency Inverse Document Frequency and Support Vector Machine 7

2.1.2 Deep Learning 8

2.1.3 Attention Mechanism 9

2.1.4 Universal Sentence Encoder 11

2.1.5 Large Language Models 11

2.1.5.1 Pretraining Tasks 12

2.1.5.2 Transformer Model 13

2.1.5.3 Bidirectional Encoder Representations from Transformers 15

2.1.5.4 Bidirectional and Auto-Regressive Transformers 16

2.1.5.5 Prompt-Enabled Language Models 17

2.1.5.6 Fine-Tuning 17

2.1.6 Meta-Learning 18

2.1.7 In-Context Learning 18

2.1.7.1 Context 19

2.1.7.2 Zero-Shot Learning 19

2.1.7.3 One-Shot Learning 20

2.1.7.4 Few-Shot Learning 20

2.1.7.5 Many-Shot Learning 21

2.2 Vulnerability Reporting 22

2.2.1 Common Vulnerability Enumeration 24

2.2.2 National Vulnerability Database 25

2.2.3 Common Vulnerability Scoring System 25

3 AI-Enabled Automated Common Vulnerability Scoring from Common Vulnerabilities and Exposures Descriptions **33**

3.1 Problem Definition 33

3.1.1 Challenges . . . 33
3.1.1.1 Poorly Described CVEs . . . 34
3.1.1.2 Human Errors . . . 34
3.1.1.3 Design Flaw in CVSS Formula . . . 36
3.1.1.4 Issues with Manual Evaluation . . . 36
3.1.2 Research Questions . . . 37
3.2 Related Works . . . 38
3.2.1 Machine Learning for the CVSS Prediction . . . 38
3.2.2 Conversion from CVSS v2.0 to CVSS v3.1 . . . 39
3.2.3 Human Expert CVSS Evaluation . . . 40
3.3 Methodology . . . 42
3.3.1 Data . . . 43
3.3.2 Tasks . . . 43
3.3.2.1 Component Classification . . . 43
3.3.2.2 Base Score Prediction . . . 44
3.3.2.3 Severity Classification . . . 45
3.3.3 Training . . . 45
3.3.4 Evaluation . . . 46
3.4 Results . . . 49
3.4.1 Predicting Common Vulnerability Scoring System 2.0 . . . 50
3.4.2 Predicting Common Vulnerability Scoring System (CVSS) 3.1 . . . 50
3.4.3 CWE-Wise Performance Analysis . . . 53
3.4.4 USE on Recent Data . . . 61
3.4.5 Resource Consumption . . . 64
3.5 Discussion . . . 66

4 Essential Aspect Labeling for Vulnerability Descriptions 70
4.1 Problem Statement . . . 70
4.1.1 What is the issue with the current vulnerability description in the CVE listing and the NVD? . . . 71
4.1.2 What are the descriptions needed for a security analyst to understand the vulnerability? . . . 73
4.1.3 What is the issue with vulnerability aspect labeling? . . . 74
4.1.4 Increase Model Size or Data Size? . . . 75
4.1.5 Research Scope . . . 75
4.2 Related Works . . . 76
4.2.1 Defining Description Quality Metrics . . . 76
4.2.2 Essential Aspects of Vulnerability Description . . . 77
4.3 Proposed Method . . . 79

4.3.1 Essential Aspect Dataset 80
4.3.2 Vulnerability Description Quality Screener 82
4.3.3 DesQS Model Performance 82
4.3.4 Essential Vulnerability Aspect Labeling Analyzer . . . 83
4.3.5 In Context Learning 85
4.3.6 Evaluation 88
4.3.7 Post-Processing 89
4.4 Result 89
4.4.1 In-Context Learning 89
4.4.2 Case Study on CVE-2024-0999 92
4.4.2.1 Zero-Shot Prediction 94
4.4.2.2 One-Shot Prediction 96
4.4.3 Impact of Essential Aspects on the Vulnerability Description Quality 97
4.4.4 Relationship Between Description Quality and CVSS . 98
4.4.5 Trends in Description Quality 101
4.4.6 CNA-Wise Description Quality 103
4.4.7 Impact of Weakness Type on Vulnerability Description Quality 105
4.4.7.1 EVAL Analysis 105
4.4.7.2 CNA Missing EVA Analysis 108
4.4.8 Running Cost Comparison 108
4.5 Limitations 110

5 Explainable Common Weakness Enumeration Mapping 111
5.1 Problem Statement 111
5.1.1 Challenges 112
5.1.1.1 Inconsistent Labels 112
5.1.1.2 Labeling Complexity 113
5.1.2 Research Question and Objectives 113
5.2 Related Work 114
5.3 Methodology 115
5.3.1 Dataset 115
5.3.2 Model and Inference 117
5.3.3 Rule Book 117
5.3.4 Evaluation 118
5.3.4.1 Group Matching for CWE Labels 118
5.3.4.2 Explanation BLEU 118
5.3.4.3 Highlight Jaccard 120
5.4 Results 120

5.4.1 ICL Setting Compare 121
5.4.2 Case Study on CVE-2021-22991 121
5.4.3 Effect of Rule Book 122
5.5 Limitations 123

6 Conclusion **131**

Appendices **134**

A Co-Authorship Form **135**

B Acknowledgements **137**

Bibliography **137**

Chapter 1

Introduction

Cybersecurity is a critical aspect of modern society, as the increasing reliance on digital technologies has led to a surge in cyber threats and vulnerabilities. With the exponential growth of vulnerabilities being discovered and reported, the evaluating process has become increasingly complex, time-consuming, and largely manual. This processing bottleneck leaves a long attack window for threat actors to exploit unpatched systems. Therefore, the ability to automatically analyze and score cybersecurity vulnerability reports is of paramount importance in enhancing the overall security posture of organizations and individuals.

Every vulnerability report typically contains a textual description of the vulnerability. This is the first information security analysts refer to when assessing the risk associated with a vulnerability. Therefore, the starting point of this research is to analyze the vulnerability descriptions and extract meaningful features that can be used for classification and severity assessment. Whether natural language alone could provide key information to automate the vulnerability analysis process is the key question this thesis aims to answer. The first task is to estimate the severity of vulnerabilities based on their descriptions. This could potentially automate the Common Vulnerabilities and Exposures (CVE) scoring process, which is currently a manual and subjective task. Next, we focus on the quality of the vulnerability descriptions. Since the descriptions

are free formed, they can vary significantly in terms of clarity, completeness, and relevance. This thesis looks into potential solutions to assess and improve the quality of vulnerability descriptions. Finally, we explore the possibility of classification of vulnerabilities based on their descriptions.

Existing vulnerability analysis works based on keywords and statistical features have shown limitations in speeding up the vulnerability analysis process and providing accurate severity estimation and classification. In the work of Sönmez [75], the author used a combination of text mining and graph theoretical analysis to classify vulnerabilities in the CVE database. The analysis was based on the frequency of keywords in the vulnerability descriptions and the relationships between vulnerability types. This approach has overly complex initial visualizations and requires manual interpretation to extract insights, which limits its scalability and efficiency since human in loop for the interpretation of the visualizations is required by design. Unsupervised learning methods such as clustering and topic modeling have also been applied to vulnerability analysis, but they often struggle to capture the nuanced relationships between vulnerabilities and their descriptions, leading to suboptimal classification performance. Neuhaus et al. [56] used topic modeling to analyze security trends in the CVE database, highlighting the challenges of capturing complex relationships between vulnerabilities. Even though faster than manual analysis, the topic modeling failed to provide accurate classification and severity estimation due to the lack of contextual understanding of the vulnerability descriptions. For three CWEs (CWE-79, CWE-89, and CWE-20), the topic modeling approach achieved 80+% in precision and recall, but for the rest of the CWEs, the performance was significantly lower, with precision and recall often below 50%. Therefore, there is a need for more advanced techniques that can better understand the context and semantics of vulnerability descriptions to improve the accuracy and efficiency of vulnerability analysis.

This thesis explores the integration of large language models (LLMs) and machine learning techniques to enhance vulnerability classification and severity

assessment. The study focuses on two key areas: (1) improving the accuracy of vulnerability severity estimation, particularly in the context of the Common Vulnerabilities and Exposures framework, and (2) refining the classification of vulnerabilities using natural language processing techniques. A major limitation in existing vulnerability classification systems is the reliance on manually curated datasets, which are often inconsistent, incomplete, or subjective. This research seeks to address these issues by leveraging LLMs to automate and improve the consistency of vulnerability analysis.

The research is guided by the following questions:

- Can large language models accurately estimate the severity of vulnerabilities based on their descriptions? Chapter 3 addresses this question by comparing the performance of a Universal Sentence Encoder (USE) model with human experts in estimating the Common Vulnerability Scoring System (CVSS) scores of vulnerabilities.

- How to assess the quality of vulnerability descriptions and its impact? Chapter 4 investigates this question by proposing a BERT-based screening model and ART-based analysis model to evaluate the quality of vulnerability descriptions and analyzing the distribution of description quality across different vendors and reporting agencies.

- To what extent can we understand the underlying weakness in a vulnerability description using machine learning techniques? Chapter 5 explores this question by developing a model to classify vulnerabilities into Common Weakness Enumeration (CWE) types based on their descriptions, providing insights into the underlying causes of vulnerabilities.

By investigating these questions, this study contributes to the broader effort of improving cybersecurity resilience. The proposed models enhanced automated vulnerability assessment methodologies, reduced reliance on manual scoring processes, and provided actionable insights for security practitioners. This thesis consists of the following contributions:

- built and compared machine learning models to estimate the severity of vulnerabilities based on the vulnerability descriptions
- compared the performance of the machine learning models with human experts on the vulnerability severity estimation task
- crafted a vulnerability description dataset and analyzed the quality of vulnerability descriptions with models trained on this dataset
- screened vulnerability database for description quality based on different vendors and reporting agencies
- analyzed the impact of description quality on the severity estimation task
- crafted a dataset for the classification of vulnerabilities with explanations and analyzed the effectiveness of the classification models trained upon this dataset

A paper on severity estimation (Chapter 3) was accepted in UbiSec 2022 as a conference paper [89]. Extended work based on this chapter has been published in the Springer's International Journal of Information Security [90]. After rigous peer review, the paper on vulnerability description (Chapter 4) came to a weak rejection. After major editorial work and method updates with ICL, the paper is ready to be submitted as a journal article. Chapter 5 addresses the root cause mapping problem requested by the MITRE's CVE program. A paper based on it is ready to be submitted to a conference.

The rest of the thesis is organised as follows: Chapter 2 introduces the prerequisite knowledge necessary to understand the rest of the thesis. Every following chapter introduces a research topic with its problem statement, related works, methodology, and result analysis as sections. Chapter 3 introduces a method to estimate the severity of vulnerabilities based on the Common Vulnerabilities and Exposures (CVE) with extendted comparison to human experts on the same task. Chapter 4 presents a work that analyzes

the quality of the CVE description and its impact on the performance of machine learning models trained on the CVE description. Chapter 5 presents on a method to classify the Common Weakness Enumeration types of vulnerabilities based on the CVE description with explanations and highlights. The last chapter draws conclusions for this PhD study and suggests future research directions.

Chapter 2

Background

This chapter provides the background knowledge necessary to understand the research presented in this thesis. It first introduces the concepts of machine learning, deep learning, and large language models, which form the foundation of the research. Then, it discusses the specific models and techniques used in this study, including the Universal Sentence Encoder (USE) and the Bidirectional Encoder Representations from Transformers (BERT). Finally, it covers the concepts in the cybersecurity domain, including the Common Vulnerabilities and Exposures (CVE) and the Common Weakness Enumeration (CWE) frameworks.

2.1 Machine Learning

Tom Mitchell defined machine learning as the study of computer algorithms that improve automatically through experience [54]. Formally, it can be stated as follows [84]:

Definition 2.1. *For a task T, machine learning is a computer program which can learn from experience E with respect to some classes of T and performance measure P where P improves with E on T.*

Machine learning forms the foundation of the research in this thesis. The primary tasks in this research include data collection and preprocessing within

the cybersecurity domain, training machine learning models, and evaluating their performance. Training a machine learning model involves feeding the model with the training data and adjusting the model's parameters to minimize the loss function as a way to optimize the model's performance. The model's performance is evaluated using the test data, which is separate from the training data. The model's performance is measured using metrics such as accuracy, precision, recall, and F1 scores.

2.1.1 Term Frequency Inverse Document Frequency and Support Vector Machine

This section introduces the Term Frequency-Inverse Document Frequency (TF-IDF) and the Support Vector Machine (SVM). Since this thesis focuses on text classification tasks, we will use the TF-IDF vectorization method to convert the text data into numerical vectors that can be fed into the classification models such as SVM.

TF-IDF is a statistical technique used to calculate the importance of a word in a document, known as the word weight [37]. TF-IDF represents a document as a numerical vector based on the statistical importance of words within a corpus, aiding feature selection for machine learning models.

The Support Vector Machine (SVM) is a supervised machine learning algorithm [57]. It conjures a hyperplane to classify the data points depending on the number of features in the dataset. The training data and the test/working data can have different types of distributions such as Gaussian or Bernoulli [57]. However, all the data should have the same set of attributes. Each SVM model has a hyperplane with a kernel function and a margin parameter. Express vectors (support vectors) form the decision boundaries within which the hyperplane centers itself with a certain distance, i.e. the margin. This margin can be 'softened' to allow exceptions to avoid overfitting. The kernel function transforms low-dimensional vectors into high-dimensional ones so that it is easier to calculate the hyperplane [57]. The choice of the kernel

function is a trade-off between the ease of the hyperplane calculation and the over-fitting of the model. The choice of the kernel function depends on the data distribution and the computational resources available. If a dataset is of high dimensionality, a linear kernel is often preferred for its computational efficiency and reduced risk of overfitting. Otherwise, a non-linear kernel such as the radial basis function (RBF) or polynomial kernel may be more suitable for capturing complex relationships in the data, albeit at the cost of increased computational complexity and potential overfitting.

SVM outperforms Naive Bayes in most TF-IDF vectorized classification tasks studied in [40] and handles nonvector data that can be a composition of multiple features of different data types. Thus, SVM serves as a strong baseline model for text classification, particularly when computational efficiency is a priority over deep learning approaches.

SVM offers a strong baseline for text classification tasks, especially when computational efficiency is a concern. While decision tree or random forest models can also be used for text classification, they may not perform as well as SVM in high-dimensional spaces created by TF-IDF vectorization. They have no contextual understanding of the text and may struggle to capture the complex relationships between words and their meanings, which can lead to suboptimal performance in text classification tasks. They also heavily rely on the manual feature engineering process, which can be time-consuming. Therefore due to the high dimensionality and complexity of the texual data, SVM is preferred over decision tree or random forest models for text classification tasks in this thesis.

2.1.2 Deep Learning

Deep learning refers to the computational models with multiple layers of neural networks that learn the representations of input data [46]. Deep neural networks optimize representations layer by layer through backpropagation, beginning at the output layer (often a sigmoid activation layer) and adjusting

weights iteratively. This representation optimization process, starting from the back/target, is known as backpropagation since the model propagates the error from the target layer back to the input layer. The trainer constructs data representations (weights and biases), known as feature maps in convolutional neural networks (CNNs), from back to front for each layer of the model.

There are three types of deep learning models: CNNs, recurrent neural networks (RNNs), and transformers, which we use the most in this thesis as in Universal Sentence Encoder and large language models. CNNs are often used for image recognition tasks; RNNs are used for sequential data tasks; and transformers are widely used for their scalability. Neural networks have the capability to approximate any function, as shown by the universal approximation theorem [34]. But, in order to achieve desired accuracy, the model needs to have sufficient data and sufficient amount of hidden units. The input and output target also needs to have a deterministic relationship [34].

2.1.3 Attention Mechanism

The Attention Mechanism in neural networks enables models to prioritize important features within an input sequence, improving the relevance of generated outputs. Equation (2.1) shows the inspiration of the Attention Mechanism from the work of Nobel Price winner Professor Daniel McFadden, who utilized this conditional probability formula to model the choice probabilities of rational decision makers [51]. The conditional probability that an individual makes choice c given their attributes $\mathbf{A}$ and the set of available options $\mathbf{O}$, as presented in Daniel McFadden's 'Conditional Logit Analysis of Qualitative Choice Behavior' is given by:

$$P(c|\mathbf{A},\mathbf{O}) = \frac{\exp\left(U_c(\mathbf{A},\mathbf{O})\right)}{\sum_{o\in\mathbf{O}} \exp\left(U_o(\mathbf{A},\mathbf{O})\right)} \tag{2.1}$$

where:

- $P(c|\mathbf{A},\mathbf{O})$ is the probability of choosing alternative c given the attributes $\mathbf{A}$ of the decision-maker and the available options $\mathbf{O}$.

- exp is the exponential function.
- $U_c(\mathbf{A}, \mathbf{O})$ is the utility of choice c, which depends on the decision-maker's attributes $\mathbf{A}$ and the characteristics of the option c in $\mathbf{O}$.
- $\mathbf{O}$ is the set of all available options.
- The denominator $\sum_{o \in \mathbf{O}} \exp\left(U_o(\mathbf{A}, \mathbf{O})\right)$ sums over all options o in the set of available options $\mathbf{O}$.

This formulation captures the dependence of the choice probability on both the decision-maker's attributes and the characteristics of the available options, consistent with McFadden's model. Later adapted in machine learning as the 'Softmax', this equation calculates the probability of a class given the target class labels, termed 'confidence logit' [44]. Since it is conveniently differentiable, a machine translation neural model utilized the same mathematical formulation to calculate the attention weights in the encoder-decoder architecture, where the authors renamed the Formula 2.1 as 'Alignment Model' [4].

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

where:

- Q is the query matrix.
- K is the key matrix.
- V is the value matrix.
- d_k is the dimensionality of the key vectors.

The Attention mechanism calculates the alignment scores by taking the dot product of the query and key vectors, scaling the attention scores by dividing them by the square root of the dimensionality of the key vectors, applying a softmax function to the scaled attention scores to obtain the attention weights,

and computing the weighted sum of the value vectors using the attention weights. The Attention mechanism allows the model to focus on different parts of the input sequence simultaneously, capturing different aspects of the input sequence and generating the final output without using the conventional recurrent neural network or convolutional neural network architectures.

2.1.4 Universal Sentence Encoder

The Bag-of-Words is a multiset of input strings. In a typical representation, the Bag-of-Words is a dictionary with the word as the key and the number of times the word appears in the input string as the value. The Deep Averaging Network (DAN) adds a neural network component that collects a feature hierarchy based on the bag-of-words representation of the input [35].

The Transformer model extends the Attention Mechanism through self-attention layers, as defined in Equation (2.2) [41] [82]. It replaces the recurrent neural network and other model architectures with Self-Attentions that are capable of recognizing interdependencies in the input feature hierarchy.

Google designed the USE model for embedding textual inputs at a sentence level instead of the prior word level to transfer knowledge among different natural language processing tasks. It consists of a DAN that embeds any textual input in a linear time and a Transformer-based encoding model that prioritizes accuracy over computational cost [12]. The USE model chooses which model to use based on the input complexity. For short sentences, it uses the Transformer model; for long sentences, it uses the DAN model.

2.1.5 Large Language Models

In this section, we provide an overview of the large language models (LLMs) that we used in our research in chronological order, starting with the foundational works of the transformer model, the Bidirectional Encoder Representations from Transformers (BERT). Then, we introduce the Bidirectional and Auto-Regressive Transformers (BART), which have optimized performance on

Table 2.1: *Popular Pretraining Tasks for Large Language Models*

Task	Definition	Supervised
MLM	Predict a randomly masked word in a sentence [**?**]	No
NSP	Predict if two sentences are consecutive [**?**]	No
LM	Autoregressively generate text [65]	No
NLI	Entail textual relations between a pair of sentences [65]	Yes
QA	Answer questions based on the context [65]	Yes
PD	Detect if two sentences are semantically equivalent [65]	Yes
LA	Determine if a sentence is grammatically correct [65]	Yes

generative tasks, and the Prompt-Enabled Language Models like the Text-to-Text Transfer Transformer (T5).

2.1.5.1 Pretraining Tasks

Pretraining involves training a machine learning model on generalized tasks before fine-tuning it for specific downstream applications. There are two types of pretraining tasks: unsupervised and supervised as shown in Table 2.1. Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) are the two unspervised pretraining tasks for the BERT model [**?**] while the Generative Pretraining Transformer (GPT) model uses autoregressive language modeling, with unsupervised and supervised tasks such as Natural Language Inference (NLI), Question Answering (QA), Paraphrase Detection (PD), and Linguistic Acceptability (LA) to improve model performance[65].

Masked Language Modeling (MLM) is a task where the model learns to predict a randomly masked word in a sentence. The goal of the MLM task is to help the model understand the context of the sentence by looking at the surrounding words, similar to the Cloze test in language learning [60] and the Continous Bag-of-Word (CBOW) model in word embedding [53]. In the work done by Devlin et al. [18], a pretraining data generator randomly selects 15%

of the words in a sentence and masks them with a special [MASK] token 80% of the time, replaces them with a random word 10% of the time, and keeps them unchanged 10% of the time. According to the authors, this random masking strategy helps the model to perform better on downstream tasks where the [MASK] token is absent in the fine-tuning data.

Next Sentence Prediction (NSP) is a task where the model learns to predict if two sentences are consecutive. The NSP task helps the model understand the relationship between two sentences, which is essential for tasks like question answering and natural language inference [**?**]. NSP prepares pretraining data by randomly selecting the first sentence from a corpus and then selecting the second sentence from the same corpus with a probability of 50% for the true next sentence and 50% for a random sentence. When compared to pretraining only with the MLM task, the NSP task boosts the performance of the BERT model on downstream tasks by 1% to 4% depending on the downstream task[**?**].

2.1.5.2 Transformer Model

Figure 2.1 shows the architecture of the transformer model. The input for the model are sequences of tokens, which are passed through an encoder, of vectors of model dimensionality. A positional encoding passes position information into the model by aggregating the token embeddings with the positional encodings. There are many kinds of positional encodings, such as the sinusoidal positional encoding [82], the absolute position [65], the learned positional encoding, and the relative positional encoding [41][55].

Figure 2.2 shows the multi-head attention mechanism in the transformer model. The model splits the input sequence into multiple segments with each having an head that learns different aspects of the input sequence. Multi-head attention concatenates the outputs of each head, which are scaled dot-product attention, and passes them through a linear layer to generate the final output.

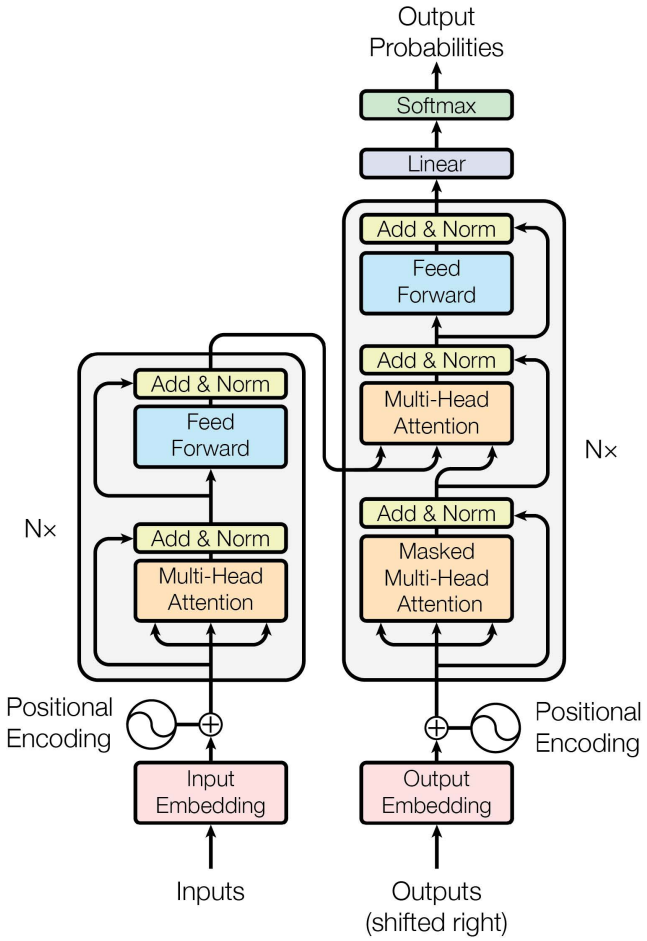


Figure 2.1: *The Transformer Model: The transformer model consists of an encoder and a decoder, each with multiple layers of self-attention and feed-forward neural networks. The encoder processes the input sequence, while the decoder generates the output sequence. The model uses multi-head attention to capture different aspects of the input sequence and positional encodings to maintain the order of the tokens. Image adapted from [82].*

Figure 2.3 shows the scaled dot-product attention used to have optionally masked multi-head attention. At first, the model calculates the attention scores by taking the dot product of the query and the key vectors. Then, the model scales the attention scores by dividing them by the square root of the dimensionality of the key vectors. An optional mask can be applied to the attention scores to prevent the model from attending to certain positions in the input sequence. Finally, the model applies a softmax function to the scaled attention scores to obtain the attention weights, which are then used to

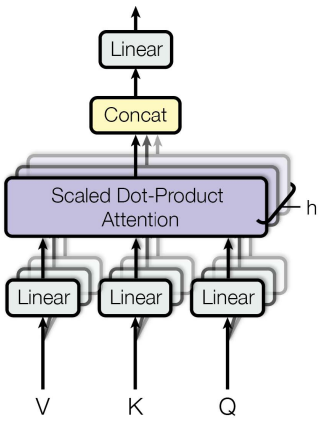


Figure 2.2: *Multi-Head Attention: The multi-head attention mechanism allows the model to focus on different parts of the input sequence simultaneously. Each head learns different aspects of the input sequence and combines them to generate the final output. Figure adapted from [82].*

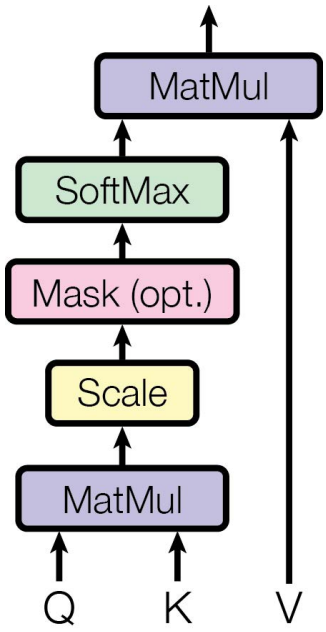
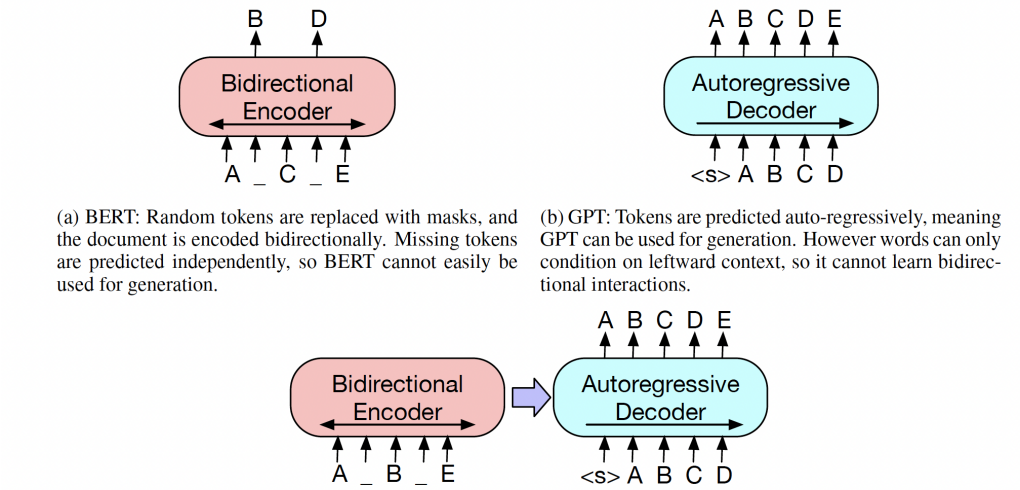


Figure 2.3: *Scaled Dot-Product Attention: figure adapted from [82].*

compute the weighted sum of the value vectors.

2.1.5.3 Bidirectional Encoder Representations from Transformers

The Google AI Language Group built Bidirectional Encoder Representations from Transformers (BERT) by pretraining a transformer model on a large



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.

(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.

(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Figure 2.4: *BART vs BERT vs GPT: BART combines BERT's bidirectional encoding power with GPT's autoregressive generation capability. Figures adapted from [48].*

corpus of text data with a masked language model (MLM) objective and a next sentence prediction (NSP) objective [**?**]. After tokenizing the BookCorpus and the English Wikipedia with a WordPiece tokenizer [73], the BERT model learns a randomly masked word by looking at the surrounding context. The NSP objective then helps the model to understand the relationship between two sentences.

However, BERT models lack domain knowledge in Cybersecurity. Due to resource constraints, the BERT model also limits the input sequence length to 512 tokens. In our case, 512 tokens are sufficient to cover the CVE descriptions. Cybersecurity-specific BERT might benefit our research by providing domain knowledge like the CySecBERT [6].

2.1.5.4 Bidirectional and Auto-Regressive Transformers

As all transformer models are in essence stacked dot-product attention layers, one may choose to specify the attention direction in the self-attention mecha-

nism [82]. Bidirectional attention allows the masked language model to learn about the past tokens and the future tokens, while autoregressive attention only allows the model to learn about the past tokens [**?**] [65]. Bidirectional and Auto-Regressive Transformers (BART) extend the BERT model by adding an autoregressive decoder used in the Generative Pretraining (GPT) model [**?**].

2.1.5.5 Prompt-Enabled Language Models

Prompt-enabled LLMs, like the Text-to-Text Transfer Transformer (T5) model [67], further expanded the capabilities of LLMs. By pretraining a large-scale language model on various tasks with meta information instructed in the input, this method enables adding a prompt for the model to guide the generation of the output. One may easily create high-fidelity output with the help of prompts. This attracted commercial attention as major companies like OpenAI, Google, and Microsoft started to provide prompt-enabled LLMs like ChatGPT [85], Gemini [80], and Copilot [14], collectively creating the Generative Model Boom.

2.1.5.6 Fine-Tuning

Fine-tuning is a machine learning technique that adapts a pretrained model to a new task by updating the model's parameters with task-specific data. As a common practice in transfer learning, where a model trained on a large dataset is adapted to a new task with a smaller dataset, it is particularly useful when the new task is similar to the original task, as the pretrained model has already learned general features that can be applied to the new task.

Natural Language Inference recognizes the textual entailment by reading a pair of sentences and deciding the relations between them, such as entailment, contradiction, or neutral [65].

2.1.6 Meta-Learning

Meta learning invokes a machine learning model with the ability to 'learning how to learn'. Instead of making model that can only solve a single task, meta learning aims to maximize the model's generalization capability. A meta learning model adapts to a new task with minimal loss in performance by enhancing the 'base learning algorithm' with the transferable knowledge from a set of auxilary tasks [13].

Meta-learning has strong connections to few-shot learning as it adapts the machine learning model to a new task with minimal fine-tuning and can handle concept drifts better than counterparts without any meta-learning capabilities. There are three types of meta-learning: metric-based, memory-based, and learning-based [26]. Metric-based learning uses a distance metric to compare the similarity between the new task and the auxilary tasks [13]. Memory-based meta learning augments the model with a memory module that stores the knowledge from external sources [25]. These memory components could be an attention-based mechanism, a RNN, or a RAG. Retrieval-augmented generation, a.k.a. RAG, acts as an external, dynamic memory, providing contextual information on-demand for the model to generate the output. By offloading the context information to external memories, RAG can reduce the model's memory footprint [49].

2.1.7 In-Context Learning

Contrary to the fine-tuning approach for learning a specific task, In-Context Learning (ICL) asks a pre-trained model to learn a new task by providing a few examples of the task without any further training or changes of the model parameters [10] [20]. This capability of LLMs is similar to how humans are able to learn a new task from analogy and demonstration [20]. In ICL, a demonstation is a pair of input and output examples that illustrate how to perform the task.

2.1.7.1 Context

In the context of the LLMs, the word 'context' refers to the input data that the model uses to generate the output. These input prompts provide the 'context' for the model to understand the generative task.

2.1.7.2 Zero-Shot Learning

Zero-shot learning refers to the task where a model can extrapolate a new class that was absent in the training data. To understand zero-shot learning, we need to introduce the concept of a semantic feature space, a semantic knowledge base, and a zero-shot classifier.

Semantic Feature Space is a n-dimensional space where each dimension represents a feature of the input pattern. These features can be categorical or continuous in nature [62]. For example, in the task of recognizing handwritten digits, the semantic feature space could contain features like the presence of a vertical line, a horizontal line, or a diagonal line.

Semantic Knowledge Base is a collection of one-to-one mappings between the feature point, in the form of a semantic space coordinate, and the class label, as a vector instead of an atomic label, learned from the training data [62].

Zero-Shot Classifier could also be called 'Semantic Output Code Classifier' as it maps the input feature pattern into a output pattern that acts like a code book in the spirit of Dietterich and Bakiri. In the task of recognizing handwritten digits, the code book contains six features, which Dietterich et al. extended to 15, with each of these features containing a binary value that indicates the presence of a certain feature in the input pattern. For example, if feature column 'vl' is 1, the input image contains a vertical line. For each class, instead of an atomic label '0' to '9', the output pattern is a 15-dimensional binary vector that indicates the presence of the 15 features in the input pattern.

Learning to recognize the handwritten digits is then equivalent to learning 15 binary classification tasks, one for each feature [19]. When optimizing a zero-shot classifier, it is better to cover more entries on this 'code book' than to cover more diverse input patterns that maps to the same output pattern [62].

In the context of ICL, zero-shot learning refers to the task where a model generates the output without any training data and in a 'off-the-shelf' condition with only pretrained parameters. Without any fine-tuning, a model in zero-shot learning setting relies solely on the prompt, which will contain the instruction, to generate the output for the task. Therefore, the quality of the output heavily depends on the quality of the prompt.

2.1.7.3 One-Shot Learning

One-shot learning refers to the task where a model can learn a new class from a single example. Before making a prediction on the test instance, there is only one training instance available for the model to learn the new class [42]. For in-context learning, there is only one demonstration of the task provided in the prompt as context.

Prone to overfitting, one-shot learning is a challenging task for the model to generalize the new class from a single example, recommened by Li et al, training examples need to be 5 to 10 for each object in the dataset [23]. Therefore, one-shot learning would often be suboptimal in the ICL setting since a good number of examples are required to learn the new task.

However, there are ways to deal with the lack of samples just like how humans can learn from a single example. For instance, a Bayesian framework could incorporate prior knowledge of the task to regularize the model [23].

2.1.7.4 Few-Shot Learning

Few-shot learning refers to a machine learning task from a limited number of examples [84]. Similar to how humans can learn from a few examples, few-shot learning tries to encapsulate the robust reasoning and analytical capabilities

of humans in the machine learning model through a data-based slow learning process and a feature-based fast learning process [74].

Few shot learning has many advantages over the other learning methods such as fine-tuning or one-shot learning. For instance, the model could learn a new task from a few examples, which is more realistic than learning from a single example. While maintaining the generalization capability of the model, few-shot learning could also reduce the requirement for the number of examples to learn the new task. Few-shot learning is cost-effective as it reduces the need for large-scale data collection and annotation, hence lowers the cost of data storage and labor involved for training a machine learning model. Compared to fine-tuning, few-shot learning requires less time and fewer hardware resources to train the model.

2.1.7.5 Many-Shot Learning

How long a context of LLM needs to be considered 'long' context? Long context refers to the ICL task where the model takes a relatively long input prompt, when compared to the last generation of LLMs, to generate the output. As shown in Table 2.2, the context size of the LLMs has been increasing over the years. Depending on this change in the context size, we can roughly divide the LLMs into different generations with different context processing capabilities. The first generation of LLMs, like BERT and GPT, has a context size of 512 tokens. The second generation, like BART and GPT-2, doubled the context size to 1024 tokens. The third generation, like GPT-3, further increased the context size to 2048 tokens. The fourth generation LLMs, like Mistral, Llama 3, and GPT-4o could process about 128,000 tokens in their context. The fifth generation LLMs feature the processing capability of what we call the long context, with the context size ranging from 1,040,000 tokens to 2,000,000 tokens. For instance, Google has made its Gemini 1.5 pro model watch the whole film Sherlock JR (1924) as the context for the task of summarizing the film [5]. This would be impossible for the previous

Table 2.2: *Context Size for Popular LLMs*

Model	Context	Year	Reference
BERT	512	2018	[18]
GPT	512	2018	[65]
T-5	512	2020	[67]
BART	1024	2019	[48]
GPT-2	1024	2019	[66]
GPT-3	2048	2020	[10]
Mistral	32,000	2023	[36]
Llama 3	128,000	2023	[21]
GPT-4o	128,000	2023	[2]
Gemini 1.5	2,000,000	2023	[81]

generations of LLMs to achieve, as their context size could only process parts of the film, ranging from a few frames to a few minutes.

What are the advantages of the Long Context LLMs? Long Context Language Models (LCLMs) eliminated the need to have two encoders that are necessary for RAG models. Instead of having a separate encoder for the context and the input, the LCLMs can process the context with all of the database information available and the input in a single encoder [47]. Since all the learning data are available in context, LCLMs can inference without extensive offline fine-tuning.

2.2 Vulnerability Reporting

The cybersecurity entities involved in the vulnerability reporting process can be categorized into three groups: governmental agencies (Agencies), product vendors (Vendors), and companies that specialize in cybersecurity (Security Firms). Each group has its own unique strengths and weaknesses in the cy-

bersecurity domain.

The United States government hold high expectations in agencies in their role as the maintainer of a worldwide database. This expectation sometimes creates a tussle between the agencies and vendors or security firms. Vendors, security firms, and agencies sometimes do not agree on vulnerabilities that have been reported. This ensuing friction often consumes the agencies' already shrinking time and resources.

As ensuring security in products is expensive it is de-prioritised by many vendors resulting in vulnerabilities. Even when vulnerabilities are discovered they may not be disclosed or patched by the vendors for various reasons such as cost of patching, fear of reputational damage, or lack of resources. Having said that, there are many organisations that have taken security seriously around the world and prioritise security spending.

The final group is the cybersecurity firms, such as, Trend Micro, Mandiant, and Cisco Talos. Security Firms typically have dedicated security analysts with decent pay [72] [16]. They serve the Vendors, occasionally Agencies, with their expertise in the cybersecurity. After realizing both the cost on their public figure and financial returns [31], Vendors would also be happy to outsource their security issues to the Security Firms [39]. However, cybersecurity is a luxury considering the cost of hiring the security firms and the shortage in the supply of cybersecurity experts [69]. Independent security researchers and bug bounty platforms also contribute to the vulnerability reporting process by discovering and reporting vulnerabilities to the vendors or agencies. They act similarly to the security firms without the legal entity. In this work, we consider these independent security researchers and bug bounty platforms same as the security firms, as they all contribute to the vulnerability reporting process by discovering and reporting vulnerabilities to the vendors or agencies.

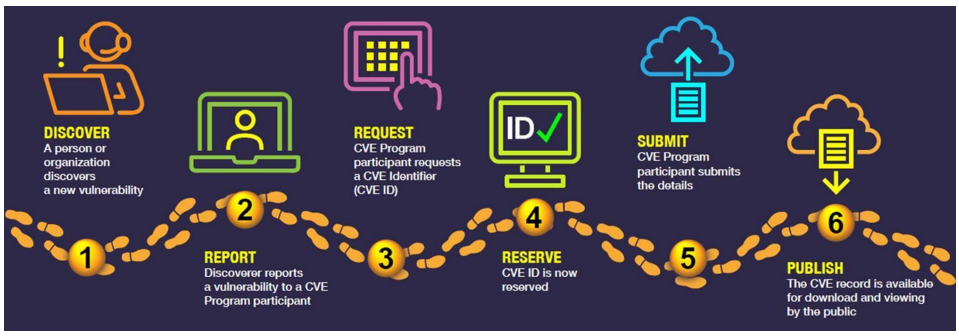


Figure 2.5: *The Six Phases of the Common Vulnerability Enumeration (CVE) Process. Figure adapted from CVE website.*

2.2.1 Common Vulnerability Enumeration

Common Vulnerability Enumeration (CVE) is a database maintained by MITRE Corp. that enlists publicly disclosed cybersecurity vulnerabilities. After October 2016, the CVE system changed from the Hub and Spoke model with MITRE as a hub to the current federated model. In the federated model, the CVE Naming Authorities (CNAs) write the CVE descriptions.

As shown in Figure 2.5, the CVE process consists of six phases: discover, report, request, reserve, submit, and publish, according to the CVE website (cve.org). Anyone may discover a vulnerability and report it to a CVE program participant. The CVE program participants, most of which are also CNAs, are major product vendors (such as Airbus, Adobe, AMD, Apple), open-source vendors, and computer emergency response teams (CERTs). These participants mostly focus on vulnerabilities reported in their products. After receiving a vulnerability report, the CNA assigns a CVE ID to the vulnerability in the request phase and the reserve phase to help track the CVE and avoid duplications. A CVE ID consists of a CVE prefix, the publication year of the CVE, and a unique number, which has four or more digits. In the Request phase, the CNA requests a CVE ID, which is reserved for the vulnerability until the CNA submits the CVE details. Finally, MITRE Corp publishes the CVE entry.

Figure 2.6: *Workflow of the National Vulnerability Database*

2.2.2 National Vulnerability Database

The NVD processes the CVE entries in a series of steps as shown in Figure 2.6. First, NVD fetches CVE data from the CVE listing. Then an NVD analyst collects as much information as possible about the CVE including using the references supplied in the CVE report and classifies the CVE to the 'type' of a vulnerability, which is the Common Weakness Enumeration (CWE) [50]. CWE is a community-developed list of software, hardware, and process weaknesses, which aims to provide a common language for discussing, categorising, and managing software weaknesses [50]. The mapping between a vulnerability in a product and its overarching weakness intends to help developers understand and mitigate the weaknesses as quickly as possible. NVD also assesses the severity of the vulnerabilities using the CVSS system, detailed in Section 2.2.3, and generates an applicability statement for each CVE entry, utilizing a logical expression XML system called the Common Platform Enumeration (CPE). Finally, NVD publishes the CVE entries with the CWE, the CVSS, and the CPE after quality assurance checks in their database [59].

The National Vulnerability Database (NVD) analyses and assesses the vulnerabilities published to the CVE. Even though the CVE program has scaled up by using the federated model, the NVD still processes the CVE entries in a centralised, linear, and labour-intensive manner as shown in Figure 2.7.

2.2.3 Common Vulnerability Scoring System

Common Vulnerability Scoring System (CVSS) is a framework for assessing the severity of vulnerabilities. Anyone on the Internet can report a vulnerability to the Common Vulnerabilities and Exposures (CVE) database maintained by the MITRE Corp. The CVE Naming Authority (CNA), typically the producer

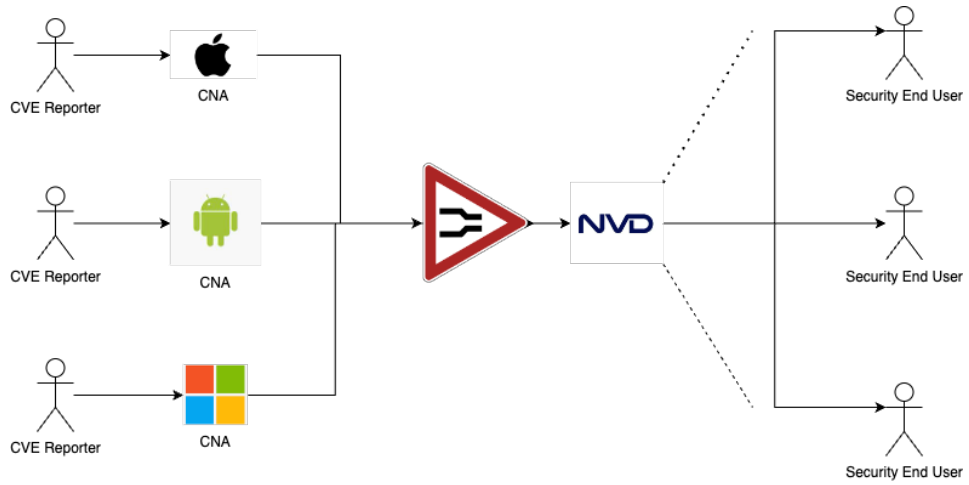


Figure 2.7: *The NVD Bottleneck: CNAs process the CVE entries in a distributed manner but the centralised processing at the NVD creates a bottleneck.*

Exploitability Metrics

Access Vector (AV)*

Local (AV:L)	Adjacent Network (AV:A)	Network (AV:N)

Access Complexity (AC)*

High (AC:H)	Medium (AC:M)	Low (AC:L)

Authentication (Au)*

Multiple (Au:M)	Single (Au:S)	None (Au:N)

Impact Metrics

Confidentiality Impact (C)*

None (C:N)	Partial (C:P)	Complete (C:C)

Integrity Impact (I)*

None (I:N)	Partial (I:P)	Complete (I:C)

Availability Impact (A)*

None (A:N)	Partial (A:P)	Complete (A:C)

Figure 2.8: *The Components of CVSS 2.0 Base Score [52]*

of the affected system, assigns each report a unique CVE identifier (CVE-ID) if the submission meets the CVE requirements. After the CVE database admits the report, the National Vulnerability Database (NVD) team at the National Institute of Standards and Technology (NIST) processes the CVE entries within a timed window, typically an hour, and assigns the CVSS to the vulnerability. Vulnerability bulletin analysts, security product vendors, or application vendors then rate the severity of a vulnerability based on the CVE description and reference links because these vendors or suppliers typically have better information about their products than the end-users, producing the CVSS Base Score. The Base Score offers a good starting point for the vulnerability discovery and patching recommendation. The end-users can craft their severity rating (CVSS Environment Score) based on the CVSS Base Score and their equipment.

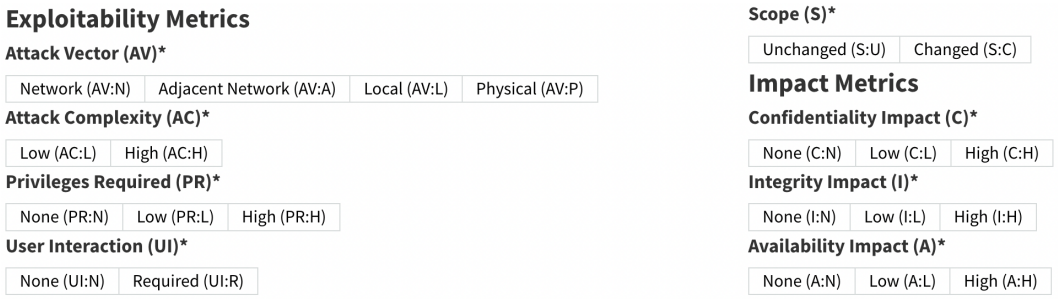


Figure 2.9: *The Components of CVSS 3.1 Base Score [24]*

There are six components in the CVSS base score version 2 (access vector, access complexity, authentication, confidentiality impact, integrity impact, and availability impact) as shown in Figure 2.8 and eight components in the CVSS base score version 3 (attack vector, attack complexity, privileges required, user interaction, scope, confidentiality impact, integrity impact, and availability impact) as shown in Figure 2.9.

The severity rating of a vulnerability depends on the base score. In CVSS version 2, the ‘low‘ severity score ranges from 0.0 to 3.9, while the ‘medium‘ severity score ranges from 4.0 to 6.9, and the ‘high‘ severity score ranges from 7.0 to 10.0. Version 3 of CVSS adds two more ratings to the severity measure: ‘none‘(0.0) and ‘critical‘(9.0-10.0). The ‘low‘ rating starts from 0.1 instead of 0.0. The ‘high‘ rating ends at 8.9 instead of 10.0 in version 2. In version 3, the ‘medium‘ severity rating uses the same base score as in CVSS v2. The CVSS 3.1 changed from three possible values (version 2.0) to two for the Attack Complexity component, leaving only the High and the Low values [24]. Attack Vector replaces the Access Vector in version 2.0. It has four values, instead of three values, with an addition of the ‘Physical‘ value, indicating the attacker must be physically present on the victim’s machine to perform attacks such as USB direct memory access [24]. Privilege Required is a new component that further categorizes the old authentication component. Along with the User Interaction component, it replaces the Authentication component in version 2.0. The User Interaction component indicates that the attacker needs misconfiguration or activity from the victim system’s users (not from the attackers)

to exploit the vulnerability. Scope indicates whether an attack changed the access rights of the infected system component [24].

```
BaseScore = round_to_1_decimal(((0.6*Impact)+(0.4*Exploitability)-1.5)*f(Impact))

Impact = 10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))

Exploitability = 20* AccessVector*AccessComplexity*Authentication

f(impact)= 0 if Impact=0, 1.176 otherwise

AccessVector     = case AccessVector of
                        requires local access: 0.395
                        adjacent network accessible: 0.646
                        network accessible: 1.0

AccessComplexity = case AccessComplexity of
                        high: 0.35
                        medium: 0.61
                        low: 0.71

Authentication   = case Authentication of
                        requires multiple instances of authentication: 0.45
                        requires single instance of authentication: 0.56
                        requires no authentication: 0.704

ConfImpact       = case ConfidentialityImpact of
                        none:             0.0
                        partial:          0.275
                        complete:         0.660

IntegImpact      = case IntegrityImpact of
                        none:             0.0
                        partial:          0.275
                        complete:         0.660

AvailImpact      = case AvailabilityImpact of
                        none:             0.0
                        partial:          0.275
                        complete:         0.660
```

Figure 2.10: *CVSS 2.0 Formula*

The CVSS formula (Figure 2.10) has received much criticism for its poor design. There is little transparency in the crafting of the CVSS formula. The National Infrastructure Advisory Council commissioned research, maintained by the Forum of Incident Response and Security Teams (FISRT), on improving the CVSS 1.0 formula. CERT, Cisco, Department of Homeland Security, MITRE Corp., eBay, IBM Internet Security Systems, Microsoft, Qualys, and Symantec participated in the making of CVSS 2.0, according to the FIRST.org website's FAQ section. Statisticians from NIST first revised the CVSS 2.0 for-

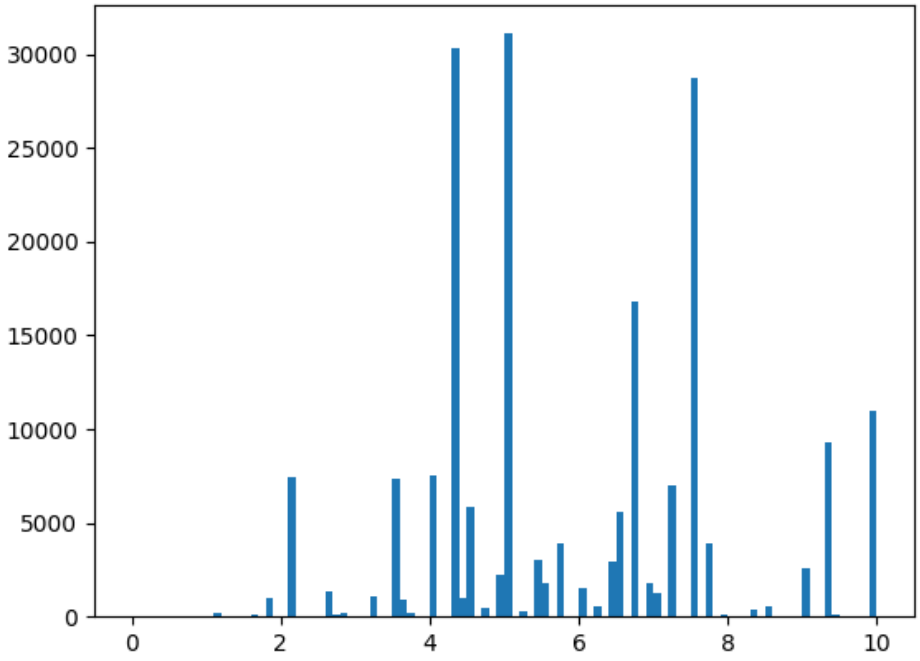


Figure 2.11: *CVSS 2.0 Base Score Distribution: the x-axis is the score in an interval of 0.1. The y-axis is the number of CVE entries. The top 3 base scores are 4.3 (30299 samples), 5.0 (29886 samples), and 7.5 (16753 samples). There are 201960 samples in total. Lots of base scores (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.1, 1.6, 2.0, 2.2, 2.5, 3.1, 3.9, 4.2, 4.5, 8.1, 8.4, 8.6, 8.9, 9.1, 9.2, 9.5, 9.6, 9.8, 9.9) have no entries.*

mula based on the feedback from the security community on the CVSS 1.0 formula. The major issue in CVSS 1.0 was its high weight on low values due to its multiplicative nature. A single low value in any component would result in a base score lower by 3 points at least. The simple stacking of multiplication caused the CVSS 1.0 to have an overwhelming 702 possible combinations. The formula crafting team split the CVSS 1.0 formula into two groups: the impact group and the exploitability group. This split reduced the number of possible combinations to two sub-vectors (27 possible combinations in the impact group and 26 possible combinations in the exploitability group). After the group split, the formula followed a series of rules for the ordering of possible

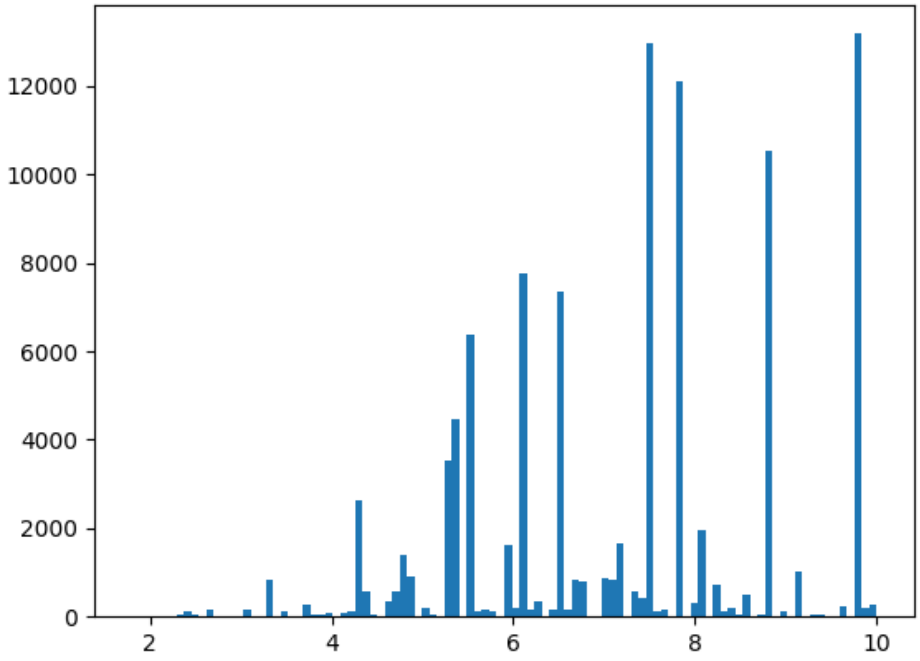


Figure 2.12: *CVSS 3.1 Base Score Distribution: The top 4 base scores are 9.8 (13185 samples), 7.5 (12951 samples), 7.8 (12120 samples), and 8.8 (10529 samples). There are 102328 samples in total. Base scores of 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 9.2, 9.5, 9.7 have no entries.*

values. However, the enforcement of these rules was not fully consistent as the SIG team reviewed and modified the ordering. After many iterations on the weighting for each group, the crafting team simplified the formula to have 0.6 and 0.4 for the impact group and the exploitability group, respectively. However, this makes the formula overflow the maximum value of 10.0. Therefore, the team added a normalization step to make sure the base score stayed within 10.0. After so many different interventions and ad-hoc solutions to cope with the formula's complexity, the CVSS 2.0 formula is still a largely black box to the security community with a skewed distribution of the CVSS 2.0 base scores (Figure 2.11).

Another criticism that the CVSS 2.0 formula received is the lack of math-

Table 2.3: *CVSS 3.1 Base Score Metrics*

Metric	Description
Attack Vector	The context of the attacker
Attack Complexity	The vulnerable conditions on the victim's end
Privileges Required	The level of privileges the attacker needs
User Interaction	Whether the victim needs to perform any actions
Scope	Whether only the vulnerable component is affected
Confidentiality	The impact on the confidentiality of the system
Integrity	The impact on the integrity of the system
Availability	The impact on the availability of the system

ematical justification. This is even true for the improved version of the CVSS 3.1 formula as well. Qualitative metrics operate with each other poorly coming to a quantitative score, the CVSS, with a small sampling space, ignoring the end user's needs in different contexts with various weights on confidentiality, integrity, and availability [77] [76]. Figure 2.11 and Figure 2.12 show the consequence of NVD's arcane mathematics: the distribution of the CVSS 2.0 and CVSS 3.1 base scores are highly skewed. Both versions of the CVSS base scores struggle to map the components of the qualitative metrics into a discrete numeric value with an interval of 0.1, from 0.0 to 10.0, wasting dozens of possible values. This ill-spread design of the CVSS formula makes it difficult to represent the severity of a vulnerability accurately. Therefore, the estimation of CVSS not only requires knowledge of the vulnerability itself but also a mathematical score distribution of this hand-crafted CVSS formula since the reference CVSS uses a vector string of metric components to create the CVSS numerical value.

For its base score, CVSS v3.1 defines the *Attack Vector, Attack Complexity, Privileges Required, User Interaction, Scope, Confidentiality, Integrity,* and *Availability* metrics as shown in Table 2.3. *Attack Vector* describes the context

(such as remote network, LAN, physical access, or system access) that the attacker uses to exploit the vulnerability. *Attack Complexity* is the vulnerable conditions on the victim's end. Low complexity means the attacker can exploit the vulnerability easily and repeatedly. High complexity means the attacker needs to perform prerequisite actions before any successful exploitation of the vulnerability. *Privileges Required* is the level of privileges the attacker needs to exploit the vulnerability. *User Interaction* describes whether the victim needs to perform any actions to expose the vulnerability. *Scope* measures if the vulnerability affects the vulnerable component only or has an effect beyond the vulnerable component. The *Confidentiality, Integrity,* and *Availability* metrics measure the impact of the vulnerability on the C.I.A. (Confidentiality, Integrity, and Availability) triad [24].

During their assessment, the NVD analyst assigns values for each of these metrics and linearly combines them into the base score of the CVSS. Security teams all over the world, when dealing with vulnerability mitigation in their environment, rely on this CVSS severity score. In this thesis we use the CVSS 2.0 and CVSS 3.1 base scores. However, CVSS 4.0 has been released. It might take a while for the NVD to update their database with the new CVSS version.

This chapter builds the knowledge foundation for the coming chapters. In the next chapter, we will explore the feasibility of using machine learning models to automate the CVSS estimation task.

Chapter 3

AI-Enabled Automated Common Vulnerability Scoring from Common Vulnerabilities and Exposures Descriptions

This chapter looks into the severity rating of the vulnerabilities in the vulnerability database. Determing the severity rating of a vulnerability is a time-consuming and labor-intensive task. Yet it is a critical task for the security teams to prioritize their resources and efforts in vulnerability mitigation. Automating this task would help the security teams to save time and resources. This chapter will not only check the feasibility of machine learning models in automating the CVSS estimation task but also compare the performance of the machine learning models with human experts.

3.1 Problem Definition

3.1.1 Challenges

This section describes the challenges in comparing the human experts and the machine learning models in the CVSS estimation task. The challenges include

bad descriptions, human errors, and the CVSS formula. These challenges make the CVSS estimation task difficult for both human experts and machine learning models.

3.1.1.1 Poorly Described CVEs

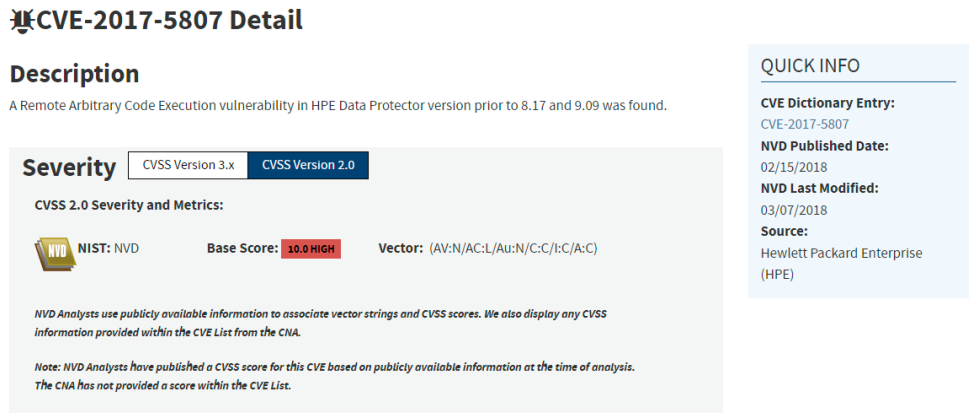
A bad description comes in many forms. It could be too short to contain enough information. For example, the description 'Buffer overflow in OpenBSD ping.' in CVE-1999-0484 is a vulnerability with a CVSS 2.0 base score of 2.1. This description only contains the vulnerability type and the affected software. It can also contain information, even though lengthy, that is irrelevant to the vulnerability.

As recommended by the NVD, a good description should contain six essential aspects: vulnerability type, root cause, affected product, impact, attacker type, and access vector [15]. However, it is often the case that one or more of these descriptive aspects are missing in the description [30] [79]. It is up to the experience of the security analyst to fill in the missing information and judge upon limited resources to estimate the CVSS score components. This can be a rather challenging task, if infeasible.

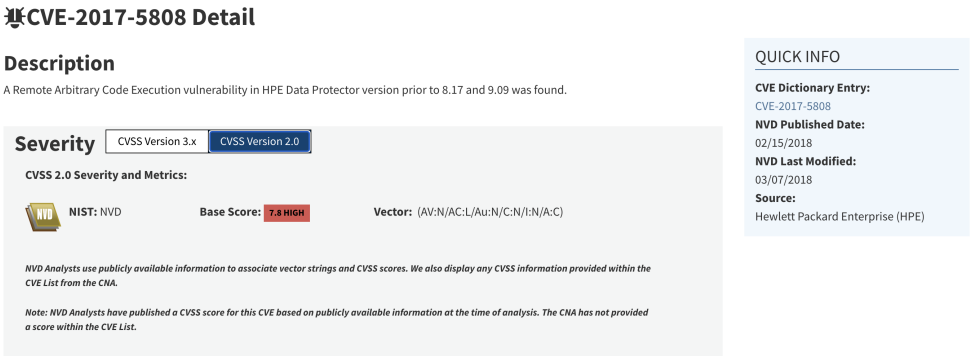
3.1.1.2 Human Errors

Figure 3.1 shows three different severity ratings for the same vulnerability. The CNA, in this case, Hewlett Packard Enterprise, reported three different CVEs for the same vulnerability. This exemplifies the human error in the CVE estimation process for unknown reasons.

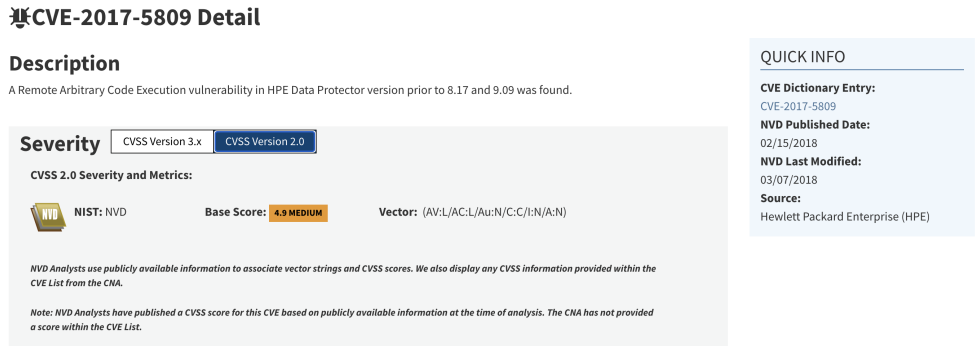
These errors in the NVD data confuse not only the AI model to learn the correct severity rating of the vulnerability but also the end-users to understand the nature of the vulnerability. While the experts have heated debates over the severity of the vulnerability, the end-users are left in the dark with extended attack windows for the malicious actors [88].



(a)



(b)



(c)

Figure 3.1: *Example of Human Error: the same vulnerabilities have three different CVSS 2.0 ratings: (a) CVE-2017-5807 with 10.0, (b) CVE-2017-5808 with 7.8, and (c) CVE-2017-5809 with 4.9.*

3.1.1.3 Design Flaw in CVSS Formula

The CVSS formula, which has received much criticism and complaints [76] [77] is largely a black box to the security community. The lack of transparency in the crafting of the CVSS formula makes it difficult for the AI model and human experts to predict the CVSS score directly.

The operators among the CVSS components are unjustified with their arbitrary weighting. This may present a challenge for the neural network or any other machine learning model to predict the CVSS score accurately.

The descriptions sometimes have some clues regarding CVSS components (for example, 'remote attacker' indicating network as the access vector), but the CVSS formula is a complex function that maps the CVSS components to the CVSS score. The eventual CVSS score sometimes is counter-intuitive even to many security experts as shown in the works of [33] with a wide spread of the CVSS 2.0 score estimation. Even though a security expert may have successfully estimated the CVSS components, the CVSS score may still be off due to the ad-hoc design of the CVSS formula.

3.1.1.4 Issues with Manual Evaluation

This section describes the issues with a manual evaluation of the CVSS score. These issues include subjectivity from human experts and the time-consuming nature of the manual evaluation.

Subjectivity

Human experts may have different opinions and interpretations, leading to subjective judgments which can cause variability and inconsistency in the CVSS analysis [33].

Time-Consuming

Evaluating a CVE manually is time-consuming. Based on data gathered up to August 7th, 2023, NVD and MITRE Corp take an average of 124 days to

progress from a CVE report to its publication of a CVSS score, with the longest waiting time recorded at 5219 days. For half of the CVEs, it takes longer than 48 days, which is the median processing time, to receive their CVSS score. Additionally, 24,548 CVEs have not been evaluated with a CVSS score for the vulnerabilities in the NVD [59].

3.1.2 Research Questions

This chapter aims to answer the following research questions:

RQ1: How effective can AI models predict the CVSS Score?

Our main research goal is to evaluate the effectiveness of AI models in predicting the CVSS score. This effectiveness measure includes the **accuracy** of the AI models for the existing data and the **cost** (time and financial) of running the AI models.

RQ2: How well do the AI models perform compared to human experts in predicting the CVSS Score?

Since machine learning models can out-learn human counterparts by training through a humanly impossible amount of data, the AI models may outperform human experts in the CVSS score prediction task with better generalization capability. However, this generalization edge is hard to measure because experts may have a lower performance due to the small testing capacity as testing on 1000 samples, which would be easy for a machine learning model, is a daunting task for a human, consuming days or even weeks.

RQ3: How long could the AI models preserve its predictive power on the CVSS score?

AI models need to be tested on unseen data to evaluate their generalization capability. It is also common for AI models to lose their predictive power over

time. Therefore, we need to evaluate AI models on unseen and more recent data to see how long they can preserve their predictive power.

3.2 Related Works

This section reviews the existing works that evaluate the severity of the vulnerability with natural language descriptions. This includes conventional machine learning models from [86] and the more recent deep learning techniques from [7] [71] [17].

3.2.1 Machine Learning for the CVSS Prediction

Yamamoto et al. [86] compared the performance of the Naive Bayes classifier, Latent Dirichlet Allocation(LDA), Latent Semantic Indexing, and supervised LDA (SLDA) to predict CVSS 2.0 scores from January 2002 to December 2013. They introduced an annual weight parameter to improve the performance of the SLDA model.

Shahid et al. [71] used a small BERT model with 4 transformer encoders to predict the CVSS 3.1 components. Costa et al. [17] compared BERT, RoBERTa, ALBERT, DeBERTa, and DistilBERT models to predict the CVSS 3.1 components with lemmatization and vocabulary addition for the tokenizer. State-of-the-art performance for predicting CVSS 3.1 components is achieved by the DistilBERT model [17].

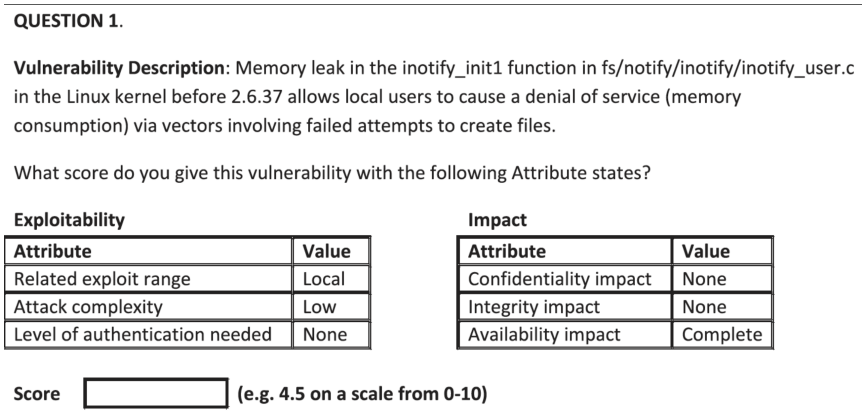
Yamamoto's work shows the potential for applying machine learning models in CVSS prediction tasks and inspires us to use a TF-IDF vectorization and a SVM model as a baseline model and CVSS score prediction as a regression task. But Yamatoto did not compare the performance of machine learning models with human experts, nor with deep learning models. Shahid's usage of BERT and Costa's comparison of the BERT family models indicate that deep learning models can predict the CVSS components. However, Shahid and Costa worked only with CVSS components or CVSS vector string predic-

tion tasks. In this chapter, we not only compare 'old school' machine learning models such as SVM with the large language models but also compare the performance of the machine learning models with human experts. We also work on models for both CVSS vector string prediction and end-to-end CVSS score prediction, without the usage of the CVSS formula.

3.2.2 Conversion from CVSS v2.0 to CVSS v3.1

There are 99,636 CVEs in the NVD database with only CVSS 2.0 scores, as of August 7th, 2023, even though CVSS 3.1 has been released since June 2019. The NVD team has been manually working on the conversion from CVSS 2.0 to CVSS 3.1 since then. The NVD database decided to discontinue the CVSS 2.0 score on July 13th, 2022 due to this ongoing workload on converting CVSS versions between 2.0 and 3.1 [59]. There is no official conversion formula for CVSS v2.0 to CVSS v3.1 [24]. Therefore, a machine learning model that can convert the CVSS v2.0 to CVSS v3.1 would be useful for the NVD team, or any cybersecurity group, to automate this conversion process.

Nowak et al. [58] collected 73,179 CVSS entries from the OWASP Vulnerability Management Center. Since the CVSS 2.0 component vectors map inconsistently to the CVSS 3.1 component vectors, they extended the CVSS 2.0 vector string with four sets of vulnerability description frequency vectors with dimensions of 50, 100, 500, and 1000. They used NLTK [9] to process vulnerability descriptions to obtain a list of keywords and divided the occurrence of these keywords by 100 to get the frequency vectors. Through machine learning algorithms, they map the seven CVSS 2.0 components into the eight CVSS 3.1 components. They achieved a median accuracy of 62.8 percent for the CVSS 3.1 score conversion task. Beck et. al [7] proposed a method to estimate the organizational risk based on the CVSS score with the help of neural network models. In their work, they hypothesized that the organizational security risks depend on the terminal device implementations. Therefore, they suggest that security officers could work bottom-up to estimate the organi-



QUESTION 1.

Vulnerability Description: Memory leak in the inotify_init1 function in fs/notify/inotify/inotify_user.c in the Linux kernel before 2.6.37 allows local users to cause a denial of service (memory consumption) via vectors involving failed attempts to create files.

What score do you give this vulnerability with the following Attribute states?

Exploitability

Attribute	Value
Related exploit range	Local
Attack complexity	Low
Level of authentication needed	None

Impact

Attribute	Value
Confidentiality impact	None
Integrity impact	None
Availability impact	Complete

Score [ ] **(e.g. 4.5 on a scale from 0-10)**

Figure 3.2: *Expert Survey Used by Holm et al. [33]: experts need to estimate the CVSS score for each vulnerability through its natural language description and its Vector String of the CVSS 2.0. This task approximates the CVSS 2.0 formula shown in Figure 2.10.*

zation's security risks. The design goal of their perceptron model is then to aid the security officers with the knowledge of the environmental context and the CVSS recursively traversing through the device topology. By recursively, Bect et. al meant to use the same perceptron model again in different cyber security contexts.

3.2.3 Human Expert CVSS Evaluation

Holm et al. [33] gathered 384 experts to evaluate 3000 vulnerabilities. These experts are from the Open Source Vulnerability Database, SCOPUS, and ten email lists from security websites such as OWASP.

Figure 3.2 shows a question used in the survey. The first section contains the natural language description of the vulnerability. The second section contains the CVSS 2.0 vector string table. The survey asks the experts to estimate the CVSS 2.0 score for each vulnerability. There are three ways to accomplish this task: taking natural language description as input, taking the CVSS 2.0 vector string as input, or taking both as input. In this human expert case, they take both the natural language description and vector string as input. This task is therefore similar to the task structure of the CVSS 2.0 formula,

which takes the CVSS 2.0 vector string as input and outputs the CVSS 2.0 score. It is an extremely challenging task for human experts as the crafting of the CVSS 2.0 formula is intransparent and unintuitive as described in the Background section 2.2.3. The survey asks the experts to give a score from 0.0 to 10.0 of 0.1 interval, which assumingly follows a normal distribution. But the reference CVSS 2.0 score spreads out in a skewed distribution as shown in Figure 2.11. It is therefore statistically challenging for human experts to estimate the CVSS 2.0 score since they would need to know this skewed distribution of the CVSS 2.0 score well enough to be an excellent human equivalent of the CVSS 2.0 formula. This 'understanding of the CVSS 2.0 formula' is a prerequisite for the human experts or the machine learning models to perform well on the CVSS 2.0 score estimation task as mentioned in the Background section 2.2.3.

There are two sets of such questions with both natural language description and CVSS vector string in the survey. The first set, which contains CVE-2012-0151, CVE-2011-2800, and CVE-2009-2203, represents different common vulnerability types. The second set contains seven randomly selected vulnerabilities from the NVD database. This small sampling size also could cause the high variance.

Holms et al. [33] grouped the result by the vulnerability CWE types. The null hypothesis for the statistical test is: 'There is no significant difference between vulnerability severity estimates by the CVSS Base Score and vulnerability severity estimates by cyber security experts'. There are two P-value choices. For CWEs with more than 100 samples, the P value is 0.05, otherwise, the P value is 0.001. We compare AI models with human performance later in the Result section for CWEs having significant P values.

Design flaws in the CVSS formula, as mentioned in the Background section 2.2.3, the small sample size of the survey, the subjectivity of human judgments, and the different opinions from experts from different backgrounds jointly contribute to the variance in the human expert CVSS estimation [33]. Nonethe-

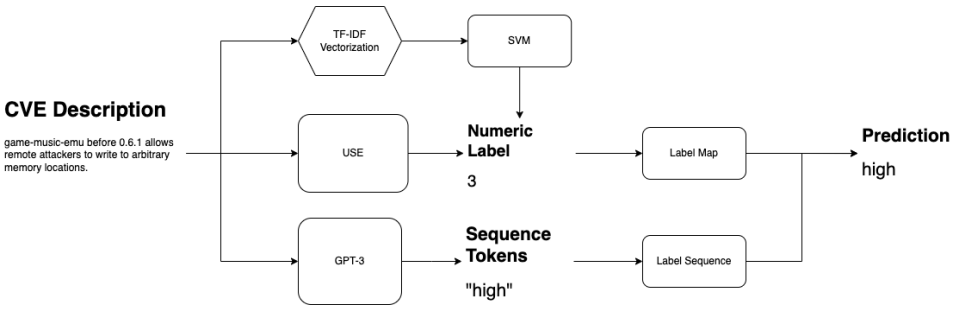


Figure 3.3: *The Proposed Methods: we propose three models to estimate CVSS scores from the natural language descriptions of the CVE entries in the NVD database. The SVM model takes a vector of TF-IDF values as input and outputs a numeric label. A post-processing script then converts the numeric label into the target prediction. The USE model predicts the same set of numeric labels and uses the same post-processing script. However, different from the SVM model, the USE model takes the natural language descriptions directly, without any pre-processing such as TF-IDF vectorization. The GPT-3 model takes in the natural language descriptions and outputs a string sequence, sometimes with noisy tokens. A post-processing script normalizes the string sequence into target labels.*

less, Holm et al. has been the only publicly accessible work to represent the performance of human experts in the CVSS estimation task. Therefore we use their work as a reference to compare the performance of AI models with human experts.

3.3 Methodology

Utilizing natural language descriptions available in the NVD, we propose three types of machine learning models, shown in Figure 3.3 to estimate the target labels of the cyber threats through natural language processing models. The natural language descriptions come from the requests submitted to the CVE listing program managed by the MITRE framework [70]. The two tasks for machine learning models are mapping these descriptions to either a CVSS

vector component or the base score. This section details the data collected for the training and the testing of these two tasks, the models used to estimate such CVSS component vectors or the base score, and the method to evaluate the performance of these models.

3.3.1 Data

The statistical analysis of the CVSS is based on data crawled from the NVD, which is available at `https://gitlab.com/zzj0402/nvd/`. The training data are 118,000 entries from the NVD database, crawled up to April 2nd, 2022 (Old Data). The test set has 51,000 entries. We set the random state to 42 to split the training and test data using the train_test_split function from the SciKit Learn library. Each entry is a JSON object with a natural language description of the vulnerability and the target labels such as severity levels, CVSS score, or a CVSS component. We collected another evaluation batch from April 2022 to July 2023 (New Data) to test the model's performance on recent data.

3.3.2 Tasks

3.3.2.1 Component Classification

CVSS metric consists of the Base Score, Temporal Score, and Environmental Score. This chapter addresses the classification task of the CVSS base score metric components. Each CVSS component classification is a multiclass classification task. The input is the natural language description of a CVE entry. The output is a CVSS component value. For example, the Access Complexity classification task for the CVSS 2.0 has the Access Complexity value(high, medium, or low) as the output.

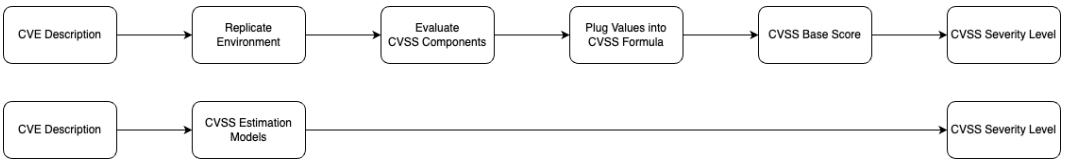


Figure 3.4: *End-to-End CVSS Estimation Workflow: the top chart shows the workflow of the CVSS severity estimation process which contains four stages [24]. The proposed deep learning models can predict the CVSS severity level from the natural language descriptions of the CVE entries. As shown in the lower chart, it is an end-to-end solution for the severity-level estimation task.*

3.3.2.2 Base Score Prediction

There are two ways for the proposed machine learning models to predict the CVSS Base Score: calculate from the components using the CVSS formula or directly output a numeric value from the natural language inputs. To calculate the Base Score from each component, one would need to build six different models, each targeting a single component for the CVSS 2.0. After such modeling, one then plugs the predicted component values into the CVSS formula to get the Base Score, ranging from 0 to 10. This is similar to the human evaluation of the CVSS score. However, the errors would aggregate on each component value. If one of the components is misclassified, the final Base Score would be incorrect. All six models need to predict the correct component values simultaneously to have the reference Base Score value.

The alternative approach is to directly output a numeric value from the natural language inputs. It is a more challenging task than the component classification-based approach. A machine learning algorithm would need to estimate not only the component values but also its formula for the numerical Base Score prediction. However, it is an end-to-end solution, which means that it requires no intermediate stages such as plugging component values into a CVSS calculation formula to get the Base Score.

3.3.2.3 Severity Classification

As a qualitative measure, the severity level is easier to predict than the numeric Base Score. Severity would also be a more intuitive measure to the user than the numeric Base Score. To calculate the severity score, the AI model takes the CVE natural language description as the input and the severity level (high, medium, and low in CVSS 2.0; critical, high, medium, low, and none in CVSS 3.1) as the output. This NLP solution eliminates the need to replicate the vulnerability environment and simplifies the workflow as shown in Figure 3.4 to avoid the usage of the complicated CVSS formula. Table 3.2 shows the classification results for the CVSS 2.0 severity levels.

3.3.3 Training

Both the GPT-3 and the USE models have four training epochs. All GPT-3 models are sequence-to-sequence, taking string inputs and outputting another string. In the classification task of CVSS 2.0 components, the input is the vulnerability description, and the output is the CVSS value string. This sequence-to-sequence setting sometimes causes the model to predict values out of the expected distribution. The USE model has a trainable Keras layer available through TensorFlow's model distribution network TensorFlowHub. For each task, the upstream model is the fifth version of the 'large' USE model with all the neural network parameters trained on the crawled NVD training dataset. A labeled map for each sample translates the string label into a numerical one. The downstream of the neural network model would have the number of predictors (a dense layer with N 'neurons' where N is the number of possible labels in the dataset). The whole model can then predict each label's likelihood as a real-valued probability score. Then a one-hot encoding indicates the index, which maps to a class label. ADAM is used as the optimizer to train all the USE models with an empirically fine-tuned learning rate. All the training losses are cross-entropy losses. All the training/evaluation accuracy is the sequence accuracy as in GPT3 metrics since the token accuracy is irrelevant to

the label, which happens to be the output sequence, typically having a single target label token.

We trained the USE model on the NVD dataset with different training and testing splits. April 2nd, 2022, is the latest date for the training dataset (2022 Split). As we suspect that the adaption of the federated CNA model in early 2017 would change the quality of the CVE descriptions, we trained the USE model on the CVE descriptions before October 2016 and after (2016 Split). As a control group, we retrained the USE model on a different dataset split on April 2nd, 2021, which is a year before our latest training split(2021 Split).

For the 2021 Split, we trained the USE model on 122079 samples. The model is tested on 61039 samples. The model diverged after 2 epochs with MSE loss as not-a-number under the learning rate of 0.0001 for the ADAM optimizer. After decreasing the learning rate to 0.00001, the model converged after 4 epochs with a MSE loss of 3.86 and a validation RMSE of 1.96. The model did not display the same learning trouble on the 18843 samples on the data from April 2nd, 2021 to August 7th, 2023. However, the USE model failed to learn any useful pattern from 2/3 of the training data as it returns a uniform prediction.

3.3.4 Evaluation

The evaluation test set is one-third of the whole dataset. For each task in the CVSS 3.1 classification task, a trained USE model predicts a test set with 28,891 samples. For the CVSS 2.0 tasks, the test set has 51,000 samples. The TensorFlow framework provides a convenient KerasLayer loading function to read the trained model into the GPU memory [1]. The model then predicts the labels for each test sample description, showing the confidence for each label. An argmax function call from the NumPy library converts the predicted probabilities into the predicted labels [32]. After the label conversion, the Classification Report tool from the SciKit Learn Metrics library calculates the precision, recall, F1-score, support, and a confusion matrix with regard

to the ground truth labels [43]. For each task, this chapter presents a classification report produced by the Scikit-Learn toolset [64]. The GPT-3 model predicts a sequence of tokens for each task, which should match the reference tokens. However, the GPT-3 model sometimes outputs tokens partially within the target vocabulary or entirely out of distribution. We handle this by selecting the relevant tokens from the predicted sequence. Suppose the model predicted 'adjacent' in the attack vector or access vector classification task. Our selection script would change 'adjacent' to 'adjacent_network' to match the reference value. If the model predicted 'adjacentnetworkconfiguration' in the same task, this selection script would change 'adjacentnetworkconfiguration' to 'adjacent_network,' removing the redundant tokens.

$$\mu = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \tag{3.1}$$

Following the same metric conventions in [33], we compare the mean error, variance, and P values of the CVSS estimations by both the human experts and the AI model. The formula 3.1 calculates the mean error. For each CVE, we add up the difference between the CVSS score predicted by the AI model ($\hat{y}_i$) and the CVSS score assigned by the NVD (y_i). Then we divide the sum by the number of CVEs (n) to get the mean error (μ).

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - \mu)^2 \tag{3.2}$$

The formula 3.2 defines the variance. For each CVE, we add up the square of the difference between the CVSS score predicted by the AI model ($\hat{y}_i$) and the mean value of all the predictions (μ). Then we divide the sum by the number of CVEs (n) to get the variance (σ^2). Python's statistics library provides a variance function to calculate the above.

Scipy implements a ttest_ind function to calculate the P value in its stats library. We use the default P-value calculation parameters with equal_var= True, nan_policy= 'propagate', permutations= None, random_state= None, alternative='two-sided', trim= 0, keepdims= False.

The CWE-Wise evaluation looks into the CWE-79 cross-site scripting vulnerability. This task compares the USE model performance against the expert's performance. We also look into the USE model prediction pattern.

We compare the AI model's performance against the expert's performance in [33] on prediction mean error, variance, and significance. The CWE order and the metrics (variance, mean value, and P value) in the figures follow conventions from [33] for the convenience of comparison.

An interesting question is how well the AI model performs on recent CVEs. These recent data test the model's ability to generalize to unseen data. We collected 31882 CVEs from April 2nd, 2022, which is the last day of our test data crawling for the USE model to test against human experts, to July 19th, 2023. We hypothesize that the AI model would perform worse on recent CVEs than the old CVEs since there is unseen information in the new CVEs. If the model performs well on recent CVEs, it indicates that it has learned the underlying pattern of the CVSS score calculation. The MITRE Corp wrote the CVEs before Oct 2016 while the CNAs wrote the CVEs after Oct 2016. To also find out whether the USE model performs better on the MITRE CVEs or the CNA CVEs, we split the data into two groups with Oct 2016 as the cutoff date. We trained the USE model on the data from April 2nd, 2021, to April 2nd, 2022, and tested the model on the data from April 2nd, 2022, to July 19, 2023.

For imbalanced data in our tasks, F1 score ignores the true negatives of a given class. F1 score is preferable to (balanced) accuracy since it can demonstrate how the model performs regarding each class. To get the overall performance of the model, we report the macro, micro, and weighted average metrics such as Precision, Recall, and F1 scores.

3.4 Results

This section presents the results of the tasks mentioned in Section 3.3.2. In Section 3.4.1 and Section 3.4.2, each classification report, adapted from the Scikit Learn Python toolkit [64], consists of label-specific and averaged metrics. There are empty cells in the accuracy row for the recall and precision columns since accuracy is a global metric, not a label-specific metric. The empty cells are not a bug or a missing calculation; they are a UI choice to indicate that Accuracy is a standalone global value, whereas the other metrics are derived from individual class performance. Each cell in the label-specific metrics has three columns(the USE model, the GPT-3 model Babbage variant, and the TF-IDF SVM model). In the lower segment of each classification report, the first row only has the accuracy in the F1-Score column. The second row has the macro-averaged (sum of the label-specific scores such as precision, recall, and f1-score, divided by the number of labels, which is 3 in component-specific metric classification tasks). The third row has the weighted average precision, recall, and F1-Score. For the CVSS score versions 2.0 and 3.1, there are classification reports for each metric component prediction task, the base score prediction, and the severity classification. The best-performing class is in bold. Overall, the USE model is the best-performing model. The GPT-3 performs head-to-head with the USE. Both deep learning models outperform the SVM model with TF-IDF vectorization.

In the later section, we present categorized CVSS 2.0 results comparing the AI model (represented by USE) against the human experts [33]. We examine the USE model capacity to generalize to unseen data, simulating the real-world scenario. We also look into the effect of different data splits on the model performance, addressing the NVD's change in its managing philosophy in 2017.

3.4.1 Predicting Common Vulnerability Scoring System 2.0

This subsection presents the results of the CVSS 2.0-related tasks. It includes the classification reports for each of the component classification tasks (Table 3.1), the Error Grid Analysis for the Base Score prediction tasks (Figure 3.5 and Figure 3.6), and the classification reports for the severity classification task (Table 3.2). The USE model performs better on the more recent test data on the Base Score prediction task with 83.5 percent of acceptable severity level predictions (Zone As in Figure 3.6) than the older test data before April 2nd, 2022 with 73.7 percent of acceptable predictions (Zone As in Figure 3.5).

As shown in Table 3.2, the USE model performs better than the Babbage variant of GPT-3 model since the USE model has a higher accuracy across the table and without out-of-distribution labels. However, the out-of-distribution prediction of the severity level labels might be due to the GPT-3 task formulation. The GPT-3 model works as a sequence-to-sequence task, while the USE model works as a multinomial classification task (labeling each description as low, medium, or high). The sequence-to-sequence task is more free-formed than the multinomial classification task as it can predict any English sentence instead of a numerical representation of the target labels.

3.4.2 Predicting Common Vulnerability Scoring System (CVSS) 3.1

This subsection presents the results of the CVSS 3.1-related tasks, which include CVSS vector string component predictions and end-to-end base score predictions. For each CVSS component prediction task, we present the classification reports with supports and label-specific precision, recall, and F1-Score. The lower section for each class report contains accuracy, macro-averaged, and weighted metrics.

Tables 3.3 3.4 3.5 present the component classification reports. Among

Table 3.1: *CVSS 2.0 Component Results: results rounded to two decimal places. Bold values indicate the best-performing model for each class. If two models have identical performance, the one with lower cost overhead is preferred.*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Access Vector Classification Report										
Adjacent	**0.83**	0.78	0.00	**0.73**	0.73	0.00	**0.78**	0.75	0.00	1779
Network	**0.97**	0.97	0.83	0.97	0.97	**1.00**	**0.97**	0.97	0.90	**42176**
Local	0.83	**0.85**	0.05	**0.85**	0.84	0.00	**0.84**	0.84	0.00	7132
Access Complexity Classification Report										
Low	**0.87**	0.87	0.68	0.88	0.89	**0.89**	0.87	**0.88**	0.77	**29858**
Medium	0.80	**0.82**	0.69	**0.82**	0.80	0.42	**0.81**	0.81	0.52	20102
High	**0.54**	0.35	0.00	0.28	**0.34**	0.00	**0.37**	0.34	0.00	1400
Authentication Classification Report										
None	0.95	**0.96**	0.84	**0.96**	0.96	0.95	**0.96**	0.96	0.89	**43265**
Single	**0.78**	0.78	0.04	0.74	**0.75**	0.01	**0.76**	0.76	0.02	7802
Multiple	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	20
Confidentiality Impact Classification Report										
Complete	0.64	**0.65**	0.10	0.65	**0.69**	0.01	0.67	**0.68**	0.01	8591
Partial	0.82	**0.83**	0.50	0.82	0.83	**0.84**	**0.85**	0.84	0.63	**25472**
None	0.87	**0.88**	0.31	0.85	**0.88**	0.14	0.86	**0.88**	0.19	17024
Integrity Impact Classification Report										
Complete	0.65	**0.70**	0.15	0.63	**0.70**	0.01	0.67	**0.70**	0.02	10100
Partial	0.77	**0.80**	0.48	**0.81**	0.80	0.48	0.79	**0.80**	0.48	**22358**
None	0.87	**0.88**	0.41	**0.88**	0.88	0.61	**0.88**	0.88	0.49	18629
Availability Impact Classification Report										
Complete	**0.72**	0.70	0.15	0.63	**0.70**	0.01	0.67	**0.70**	0.02	10100
Partial	0.77	**0.80**	0.48	**0.81**	0.80	0.48	0.79	**0.80**	0.48	**22358**
None	0.87	**0.88**	0.41	**0.88**	0.88	0.61	**0.88**	0.88	0.49	18629

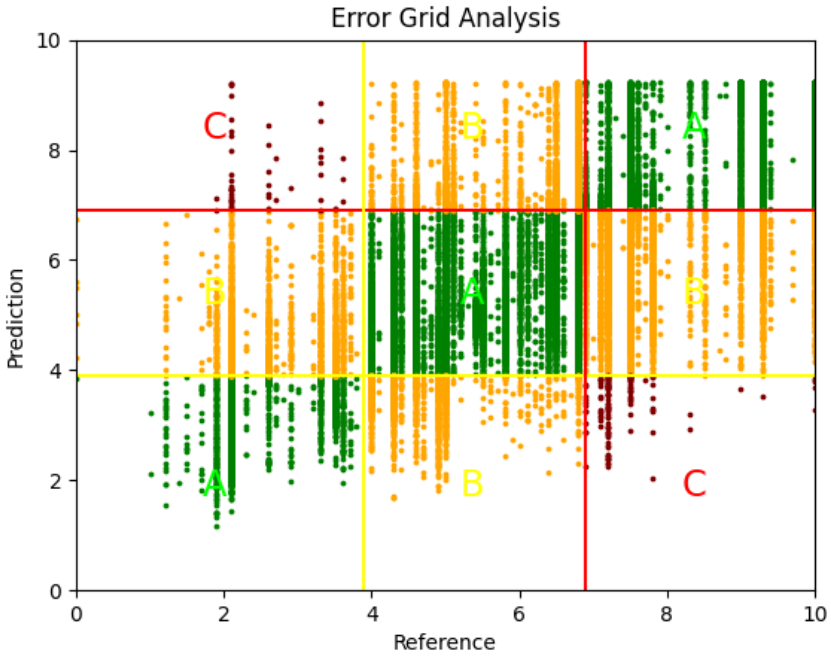


Figure 3.5: *Error Grid Analysis of the USE Model for CVSS 2.0 Base Score Prediction: the error grid consists of a 'medium' threshold line (3.9, yellow) and a 'high' threshold line (6.9, red). The three Zone As (colored in green, 73.7 percent) in the grid indicate acceptable predictions within the same level of severity as the reference values. The four Zone Bs (colored in orange, 25.9 percent) indicate predictions that are one level away from the reference severity level. The two Zone Cs (colored in red, 0.4 percent) indicate predictions that are two levels away from the reference severity level.*

these models, the Universal Sentence Encoder performs best in all the classification reports. The GPT-3 Babbage performs close to the USE model, none significantly. The TF-IDF SVM Model performs well in the recall metric across tasks. Table 3.6 shows that the Universal Sentence Encoder has the best overall performance in the Severity classification task based on the accuracy, the macro-averaged F1 score, and the weighted F1 metrics.

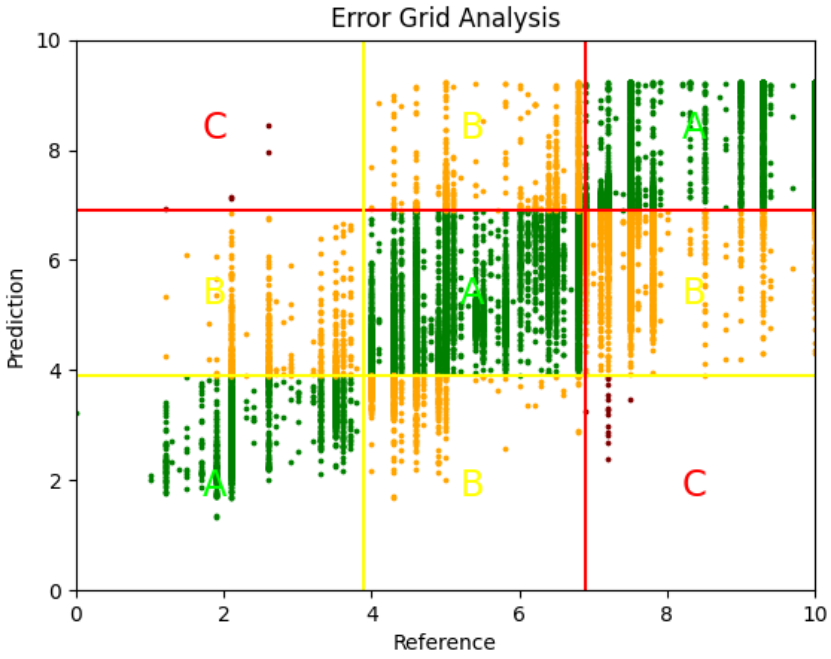


Figure 3.6: *EGA of the USE for CVSS 2.0 Base Score Prediction on New Data: the Zone A (acceptable) 83.5 percent; Zone B (error by one severity level) 16.4 percent; Zone C (error by two severity levels) 0.06 percent. There are 31,882 CVEs in the new test dataset, collected after April 2nd, 2022. The most frequent score, 9.2, occurred 329 times in the test predictions.*

3.4.3 CWE-Wise Performance Analysis

This section presents the CWE-wise performance analysis of the models. We first grouped all the test data by their vulnerability type. Then we evaluated the trained models on the test data according to CWE types. For the ease of comparative analysis between AI models and human experts, we ordered the CWEs in the ranking conventions from the paper [33]. We suspected CVE description length has a significant impact on the prediction error and investigated three categories of errors made by the USE model: critical errors, medium errors, and correct predictions. Critical errors are predictions with a CVSS raw error of 3 or more. Medium errors are predictions with a CVSS raw error between 0.1 and 3. Correct predictions are predictions on the base score with exact matching of the reference CVSS.

Table 3.2: *The Classification Report (Following Conventions from SciKit Learn) of the Severity Level Prediction*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Low	**0.64**	0.59	0.00	**0.61**	0.58	0.00	**0.62**	0.59	0.00	5271
Medium	**0.80**	0.79	0.59	0.84	0.80	**0.95**	**0.82**	0.80	0.72	30053
High	**0.76**	0.72	0.24	**0.71**	0.71	0.04	**0.73**	0.71	0.06	15763
accuracy							**0.77**	0.75	0.57	51087
macro	**0.73**	0.35	0.27	**0.72**	0.35	0.33	**0.72**	0.35	0.26	51087
weighted	**0.77**	0.75	0.42	**0.77**	0.75	0.57	**0.77**	0.75	0.45	51087

CWE-79 is the most common CWE in the dataset with 5935 samples as shown in Table 3.7. According to MITRE.org, CWE-79 categorizes vulnerabilities with 'Cross-site Scripting' or XSS issue. Improper neutralization of input during web page generation makes a web application vulnerable to malicious script injection. Due to XSS's popularity in the NVD dataset, we analyzed the prediction errors of CWE-79 in detail. We assumed that the description quality of the CWE-79 typed vulnerabilities could represent the overall description quality of the NVD dataset and randomly picked 9 samples from each category and analyzed the description length and the prediction error.

There are no strong correlations between the length of the description and the prediction error since the model is equally capable of predicting the CVSS score for the short and the long descriptions.

There are 5739 CVEs with the word 'XSS' in the test data. The AI model tends to underestimate the CVSS score for the XSS vulnerabilities. Within these samples, the model underestimates 4618 samples with a mean error of 0.47 around a mean CVSS of 3.88 with a deviation of 0.289. The model overestimates 1121 samples with a mean error of 0.47 around a mean CVSS of 4.04 with a deviation of 0.4.

The human experts' data come from [33]. Table 3.7 shows the sample size

Table 3.3: *CVSS 3.1 Component Results (Part 1)*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Attack Complexity Classification Report										
Low	**0.96**	0.96	0.93	0.98	0.97	**1.00**	**0.97**	0.97	0.96	26842
High	**0.76**	0.57	0.23	**0.71**	0.53	0.00	**0.73**	0.55	0.00	2049
Accuracy							**0.95**	0.94	0.93	28891
Macro	**0.82**	0.77	0.58	**0.75**	0.75	0.50	**0.78**	0.76	0.48	28891
Weighted	**0.94**	0.94	0.88	**0.95**	0.94	0.93	**0.94**	0.94	0.90	28891
Attack Vector Classification Report										
Network	**0.94**	0.93	0.74	0.94	0.93	**0.98**	**0.94**	0.93	0.84	21051
Adjacent	**0.66**	0.54	0.63	0.63	**0.65**	0.02	**0.65**	0.59	0.03	770
Local	**0.83**	0.81	0.56	**0.84**	0.80	0.09	**0.83**	0.80	0.16	6727
Physical	**0.72**	0.62	0.00	**0.59**	0.58	0.00	**0.65**	0.60	0.00	343
Accuracy							**0.90**	0.89	0.73	28891
Macro	**0.79**	0.73	0.48	**0.75**	0.74	0.27	**0.77**	0.73	0.26	28891
Weighted	**0.90**	0.89	0.69	**0.90**	0.89	0.73	**0.90**	0.89	0.65	28891
Availability Impact Classification Report										
None	**0.91**	0.88	0.49	0.85	**0.88**	0.54	**0.88**	0.88	0.51	11194
Low	**0.70**	0.30	0.00	0.29	**0.34**	0.00	**0.41**	0.32	0.00	697
High	0.89	**0.91**	0.66	**0.95**	0.90	0.64	**0.92**	0.90	0.65	17000
Accuracy							**0.89**	0.88	0.58	28891
Macro	**0.83**	0.70	0.38	0.69	**0.71**	0.39	**0.73**	0.70	0.39	28891
Weighted	**0.89**	0.88	0.57	**0.89**	0.88	0.58	**0.89**	0.88	0.58	28891

Table 3.4: *CVSS 3.1 Component Results (Part 2)*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Confidentiality Impact Classification Report										
None	**0.83**	0.80	0.47	**0.83**	0.82	0.19	**0.83**	0.81	0.27	6365
Low	**0.87**	0.80	0.81	0.79	**0.80**	0.22	**0.83**	0.80	0.34	5507
High	**0.89**	0.89	0.63	**0.92**	0.88	0.92	**0.90**	0.89	0.75	17019
Accuracy							**0.87**	0.85	0.63	28891
Macro	**0.86**	0.83	0.64	**0.84**	0.83	0.44	**0.85**	0.83	0.45	28891
Weighted	**0.87**	0.85	0.63	**0.87**	0.85	0.63	**0.87**	0.85	0.57	28891
Integrity Impact Classification Report										
None	**0.86**	0.60	0.47	**0.89**	0.42	0.46	**0.88**	0.50	0.46	9034
Low	**0.87**	0.68	0.88	0.83	**0.75**	0.21	**0.85**	0.71	0.34	4971
High	**0.90**	0.73	0.58	**0.90**	0.84	0.73	**0.90**	0.78	0.65	14886
Accuracy							**0.88**	0.69	0.56	28891
Macro	**0.88**	0.67	0.64	**0.87**	0.67	0.47	**0.87**	0.66	0.48	28891
Weighted	**0.88**	0.68	0.60	**0.88**	0.69	0.56	**0.88**	0.68	0.54	28891
Privilege Required Classification Report										
None	0.88	**0.89**	0.69	0.92	0.87	**0.96**	**0.90**	0.88	0.80	18707
Low	**0.76**	0.69	0.59	0.72	**0.74**	0.19	**0.74**	0.71	0.29	8033
High	**0.73**	0.60	0.33	**0.59**	0.59	0.02	**0.65**	0.59	0.04	2151
Accuracy							**0.84**	0.81	0.68	28891
Macro	**0.79**	0.44	0.54	**0.74**	0.44	0.39	**0.76**	0.44	0.38	28891
Weighted	**0.84**	0.81	0.64	**0.84**	0.81	0.68	**0.84**	0.81	0.60	28891

Table 3.5: *CVSS 3.1 Component Results (Part 3)*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Scope Classification Report										
Unchanged	**0.97**	0.97	0.87	0.99	0.83	**1.00**	**0.98**	0.97	0.93	24156
Changed	0.92	0.86	**0.93**	0.84	**0.97**	0.21	**0.88**	0.84	0.35	4735
Accuracy							**0.96**	0.95	0.87	28891
Macro	**0.95**	0.92	0.90	**0.92**	0.90	0.60	**0.93**	0.91	0.64	28891
Weighted	**0.96**	0.95	0.88	**0.96**	0.95	0.87	**0.96**	0.95	0.83	28891
User Interaction Classification Report										
None	**0.94**	0.94	0.74	**0.96**	0.95	0.96	**0.95**	0.94	0.83	19088
Required	**0.92**	0.89	0.80	**0.89**	0.87	0.34	**0.90**	0.88	0.48	9803
Accuracy							**0.93**	0.92	0.75	28891
Macro	**0.93**	0.92	0.77	**0.92**	0.91	0.65	**0.93**	0.91	0.66	28891
Weighted	**0.93**	0.92	0.76	**0.93**	0.92	0.75	**0.93**	0.92	0.71	28891

Table 3.6: *Severity Level Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Low	**0.56**	0.25	0.00	**0.24**	0.33	0.00	**0.34**	0.28	0.00	517
Medium	0.76	**0.77**	0.51	**0.81**	0.77	0.53	**0.78**	0.77	0.52	11455
High	**0.73**	0.72	0.50	**0.73**	0.70	0.67	**0.73**	0.71	0.57	12474
Critical	**0.67**	0.60	0.28	0.57	**0.61**	0.01	**0.62**	0.60	0.02	4445
Accuracy							**0.73**	0.71	0.50	28891
Macro	**0.68**	0.59	0.32	0.59	**0.60**	0.30	**0.62**	0.59	0.28	28891
Weighted	**0.73**	0.71	0.46	**0.73**	0.71	0.50	**0.73**	0.71	0.45	28891

Table 3.7: *Sample Size for Each CWE, AI vs Human: CWE-255, CWE-16, and CWE-59 have significant P values.*

CWE	AI	Human
CWE-79	5935	194
CWE-310	1222	14
CWE-362	294	6
CWE-200	2020	373
CWE-264	1647	122
CWE-255	234	18
CWE-16	88	10
CWE-59	227	15
CWE-22	1381	71
CWE-287	877	26
CWE-78	743	3
CWE-352	1050	30
CWE-94	834	66
CWE-89	2304	213
CWE-399	856	88
CWE-189	378	41
CWE-20	2682	413
CWE-119	4364	513
CWE-134	87	4

of both the USE model test data and the human data for each CWE. Due to the manual effort required to collect the survey data, the sample size for human experts is much smaller.

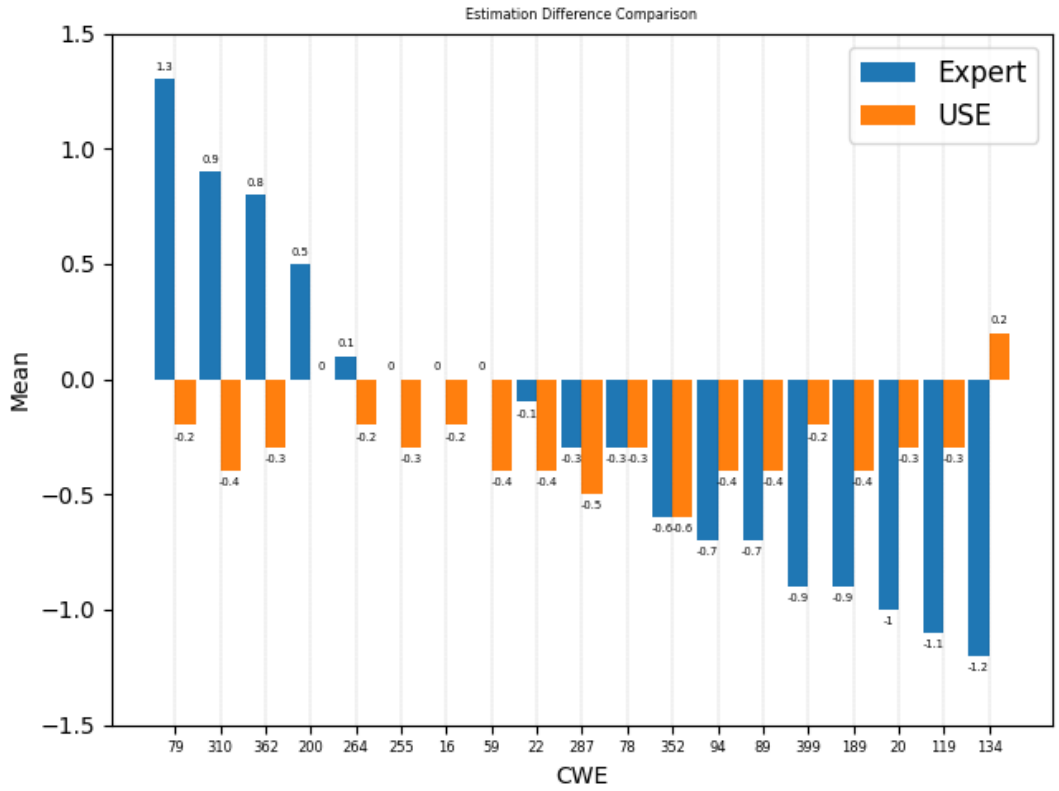


Figure 3.7: *Estimation Mean Error Comparison Between the AI Model and the Human Experts. In general, the AI model underestimates the impact of the vulnerability CVSS. AI outperforms the human experts except on CWE-264, CWE-255, CWE-16, and CWE-59.*

The AI model outperforms the human experts by a large margin. The USE model has a macro mean squared error of 0.128, which is 4 times lower than the human experts (0.537). Both the human and the AI can estimate CVSS of the Cross-Site Scripting (XSS, CWE-79) vulnerabilities with significant accountability with both having near 0 P-values. The AI model (-0.18 mean error) estimates about 7 times more accurately than the human experts (1.3 mean error), with 14 percent of the human error magnitude. The AI (0.2 variance) predicts about 16 times more stable than the human experts (3.3 variance) with 6 percent of the estimation swing. Human experts sampled 194

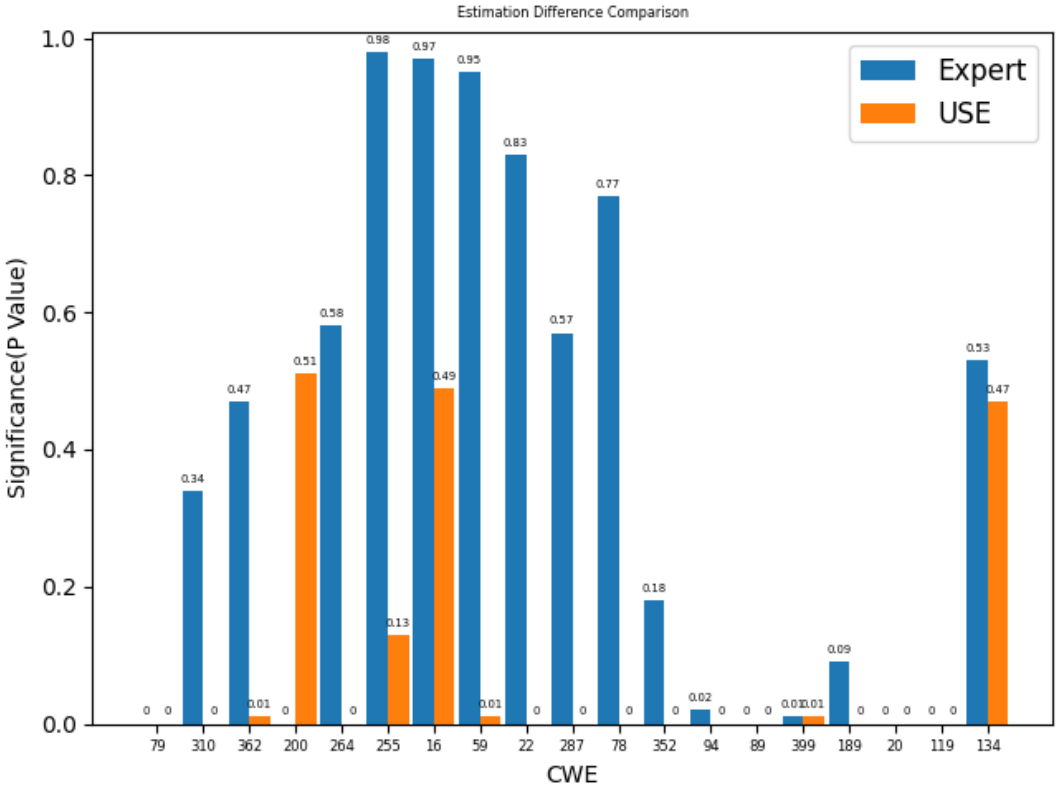


Figure 3.8: *Estimation Significance Comparison The AI model predicts with more significant results with lower P-Values than human experts except on CWE-200.*

CVEs which is 3 percent of the AI's workload (5935 samples). 9.8 percent of the tested data of the AI model is of CWE-79 while the human expert data has 6.4 percent. In the XSS-typed vulnerability evaluation task, the AI model achieved 7 times lower mean error and 16 times lower variance than the human experts.

However, the AI model underestimates the severity of the vulnerability CVSS. Figure 3.7 and 3.9 compare the CVSS base score prediction power between the AI model and the human experts. Overall, the USE model tends to underestimate the CVSS score and estimates with fewer errors than the human experts on all the CWEs except CWE-264, CWE-255, CWE-16, and CWE-59. Compared to the AI model, human experts perform with higher variance except on test data from CWE-362 and CWE-287. CWE-200 vulnerabilities, particularly, have low significance for AI model predictions.

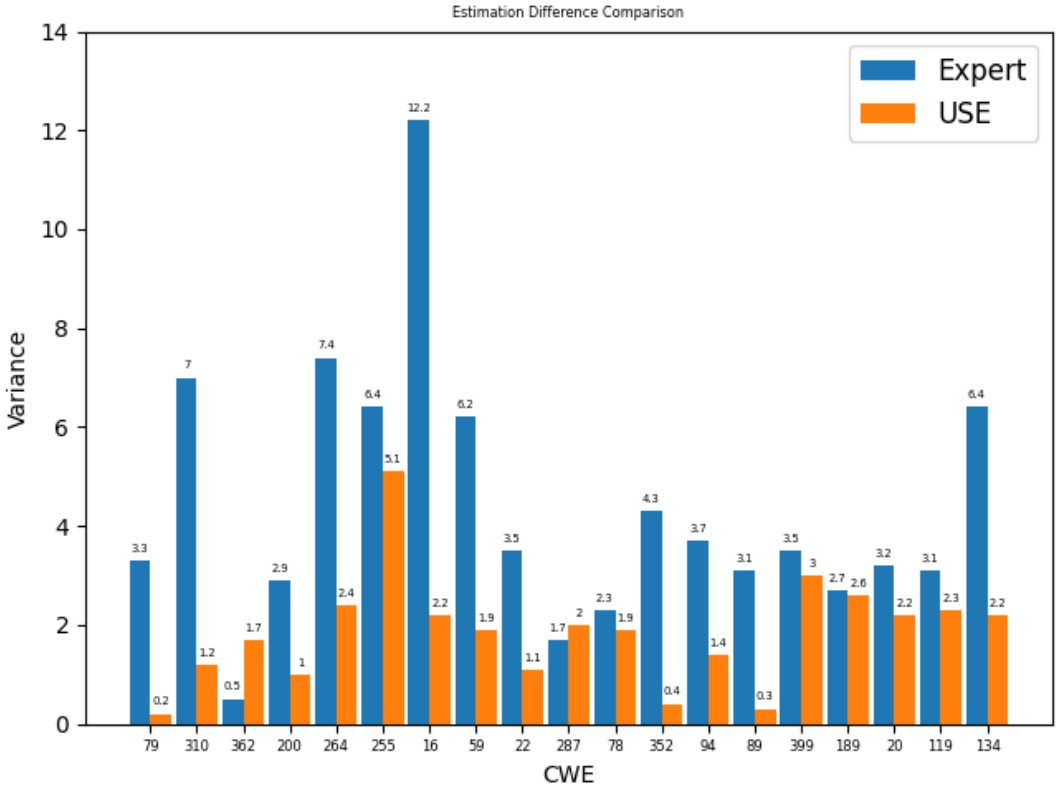


Figure 3.9: *Estimation Variance Comparison. The AI model has lower prediction variance than human experts except on CWE-362 and CWE-287.*

On these significant results (P value smaller than 0.05), the best performing category is in CWE-79. The USE model has a mean error of -0.2, which is more than 6 times lower than the human experts (mean error 1.3), and a variance of 0.2, which is 16 times lower than the human experts (variance 3.3). For CWE-94, the USE model performs the worst with a mean error of 0.4, which is still almost half of the human experts (mean error 0.7), and a variance of 1.4, which is less than half of the human experts (variance 3.7).

3.4.4 USE on Recent Data

This section examines the performance of the AI model for the CVSS estimation task on newer data. We trained the USE model on the data before April 2nd, 2022 (Old Data), and tested the model on the data from April 2nd, 2022 till July 2023 (New Data, Recent Data). We also examined the effect of CVE management philosophy shift and different data splits (2021 split, in-

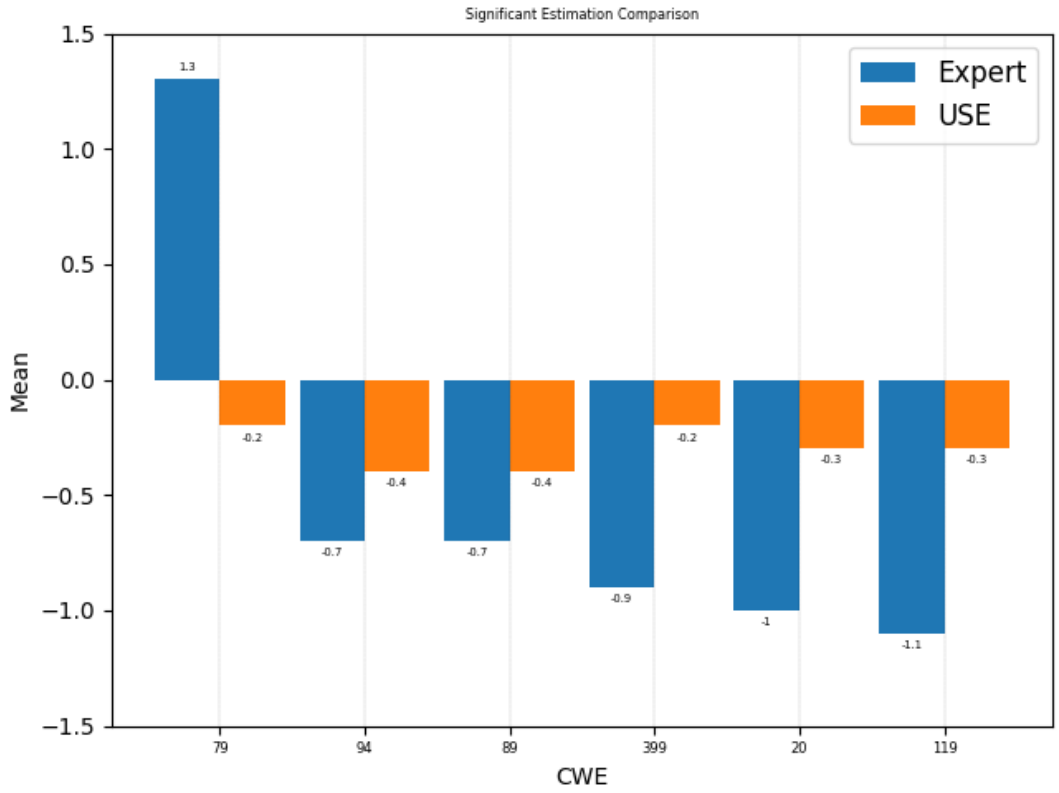


Figure 3.10: *Significant Estimation Comparison: the AI model outperforms the human experts in CWEs with significant P-Values (smaller than 0.05). AI models underestimate the CVSS value while human experts do the same except with CWE-79.*

stead of 2022 split above) on the model performance. The macro-averaged mean squared error of the USE model on the recent data is 0.118, which is a 7.8 percent improvement over the test data before April, 2nd, 2022. This comparison of model performance on new and old data shows that the model has a strong generalization ability for unseen vulnerabilities. As seen in Figure 3.12, the model performs similarly on the new data as on the old ones.

Figure 3.14 indicates that the model has fewer significant results on CWE-362, CWE-255, CWE-59, and CWE-78. As a comparison, CWE-362, CWE-59, and CWE-78 are significant in the old data before April 2nd, 2022, with P values smaller than 0.05. For CWE-362, there are 149 samples in the new dataset. There are 294 samples in the old dataset. For CWE-59, there are 158 samples in the new dataset and 227 samples in the old dataset. For CWE-78,

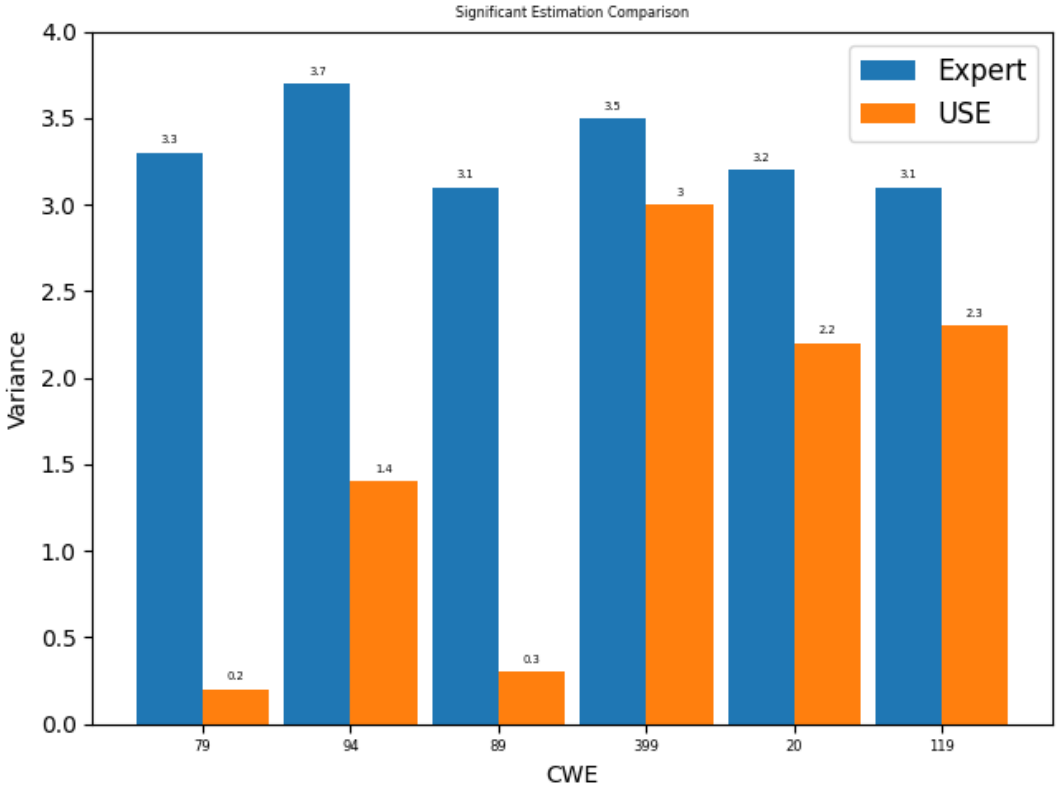


Figure 3.11: *Significant Estimation Comparison: the AI model outperforms the human experts.*

there are 124 samples in the new dataset and 743 samples in the old one. One plausible cause explaining insignificant results in new data is the insufficiency of the data.

Since MITRE Corp. has shifted the management philosophy of the CVE program from centralized management to decentralized management, the vulnerability description quality has potentially improved. To investigate this hypothesis, we tried the USE model on two data subsets: the data before the CVE paradigm shift and the data after the shift. The USE model has an MSE of 1.22 on the MITRE-managed data and 1.31 on the CNA-managed ones. The USE model diverged during the training process for the 2021 split (NVD data till April 2021) with a learning rate of 0.0001 for the ADAM optimizer. After lowering the learning rate 10 times to 0.00001, the model converged with an MSE of 3.85 and root mean squared error (RMSE) of 1.96 in the fourth epoch. On the data that are from April 2021, the model has an MSE of 3.63

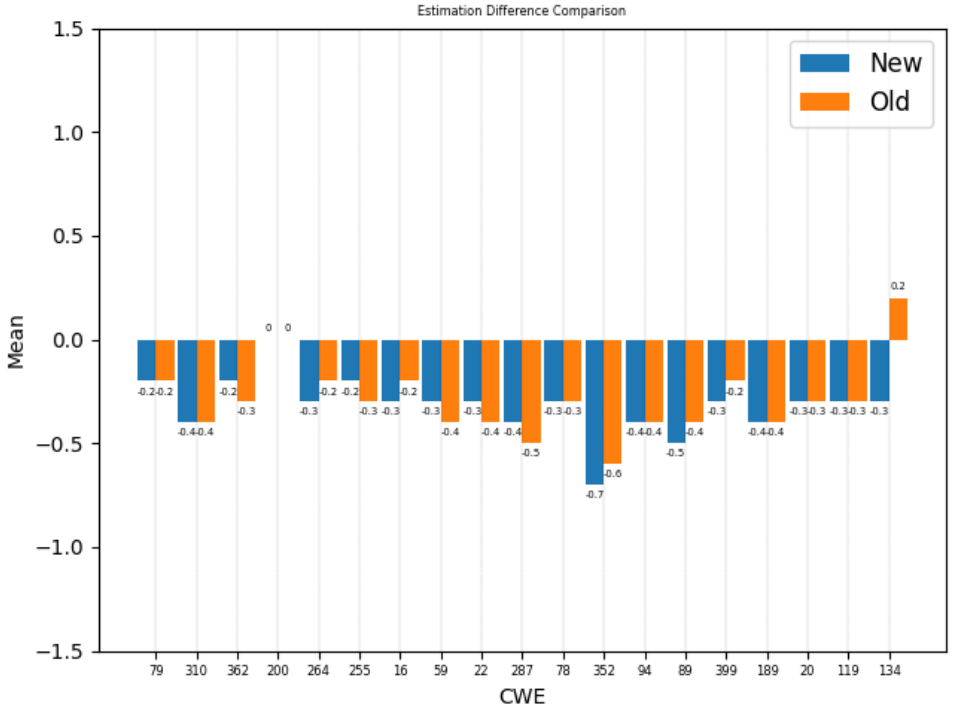


Figure 3.12: *Prediction Mean Error Comparison on the Recent Data: in general, the AI model underestimates the impact of new vulnerabilities. New data are collected from April 2nd, 2022 to July 2023. Old data I collected before April 2nd, 2022.*

and RMSE of 1.91. However, the model failed to predict any meaningful results on the data after April 2021 as it predicts a uniform number for all the samples.

The 2021 model (USE trained on the data before April 2021) has an MSE of 2.79 on its validation data (one-third of the data before April 2021, randomly selected). On the test data after April 2021, the 2021 model has an MSE of 2.17. This result shows that the model has a strong predictive ability on the recent data. Or, it is easier to predict the CVSS score for the recent data than the old data.

3.4.5 Resource Consumption

Table 3.8 shows the resource consumption to train and test USE, SVM and BERT models on a P100 instance on Kaggle with 30699 test inferences and

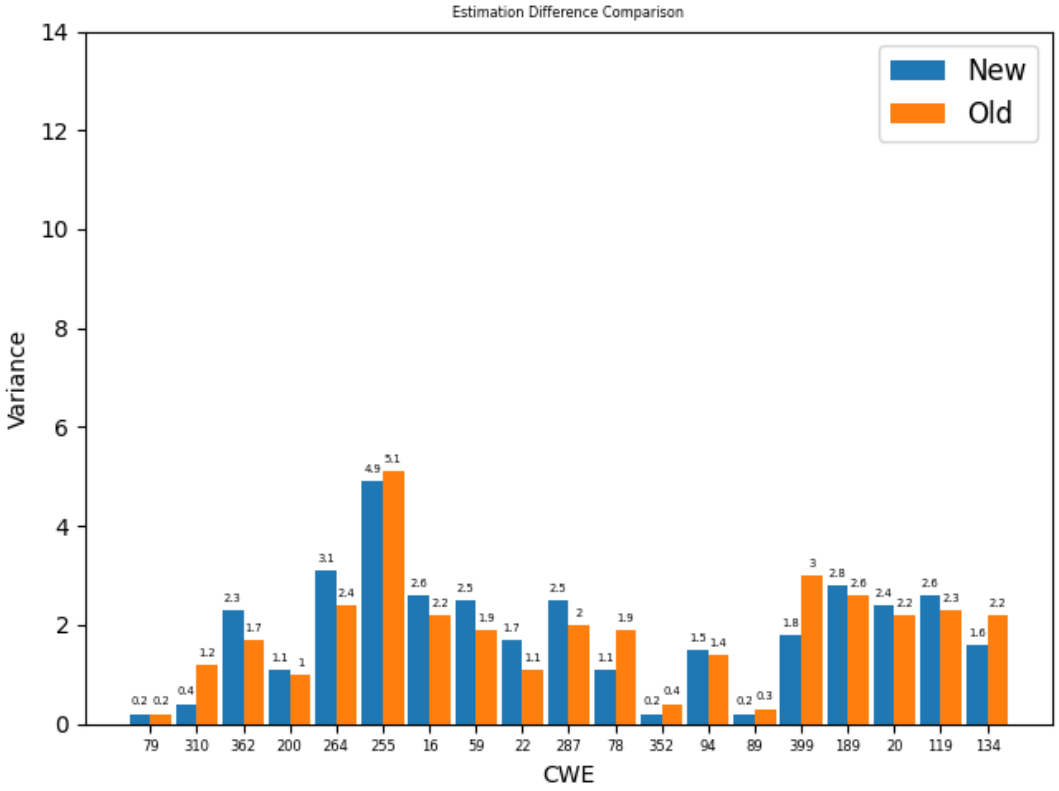


Figure 3.13: *Prediction Variance Comparison on the Recent Data*

92097 training samples of one epoch on a CVSS prediction task with a batch size of 9. The USE-V2 model consumes 3.06 GB of video memory and 1.04 GB of storage, which is twice as much as the USE-large variant, which peaks at 3.66 VRAM consumption. However, the USE-V2 model trains and inferences 6x faster than the USE-large model.

The GPT-3 model runs remotely on the OpenAI API therefore there is no available data on its resource consumption. BERT, which represents the large language model, consumes 7.11 GB of video memory and 0.42 GB of storage, taking 75 minutes to train and test one epoch with a batch size of 9. The BERT model has the best performance with 0.72 MAE.

The SVR model with the TF-IDF vectorization consumes 225 minutes to train and requires no video memory and runs on the CPU instance (Intel Xeon 2.20 GHz four vCPU cores with 32 GB main memory). Even though the SVR has the best performance (0.74 CVSS base score) in terms of the mean absolute error (MAE), the USE model is more efficient in terms of time required to train

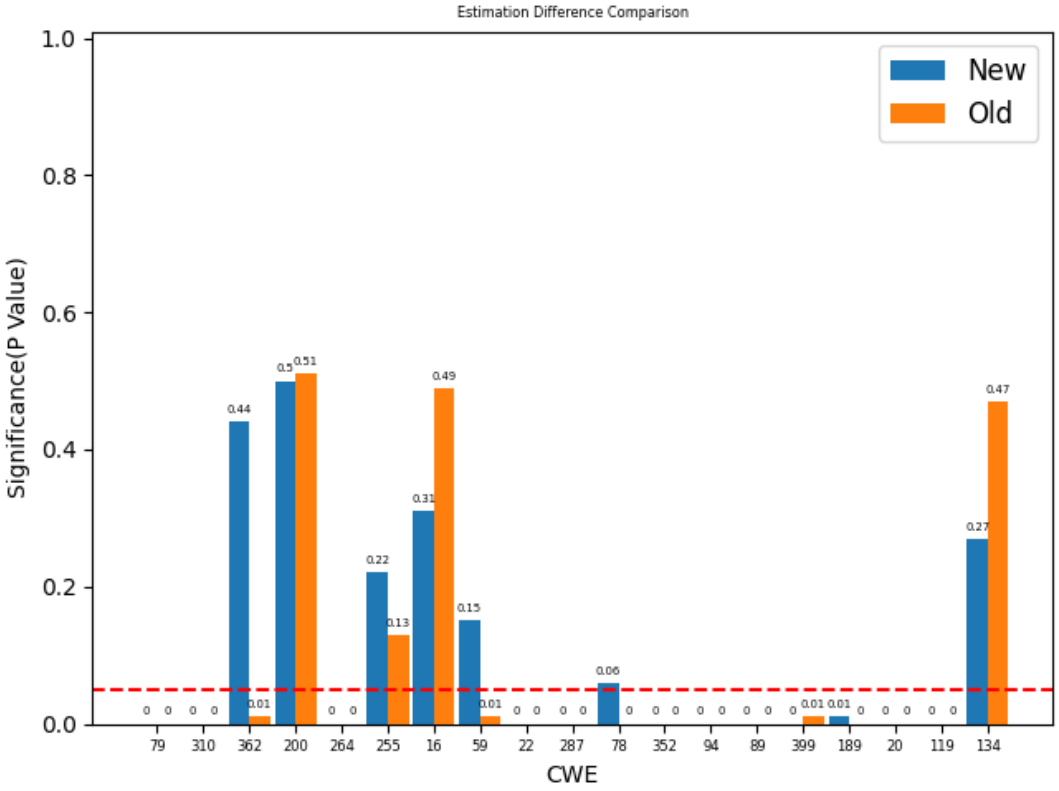


Figure 3.14: *Prediction Significance Comparison (Lower is Better) on the Recent Data: The model performs comparably on the new data except on CWE-362. There are 149 samples in the new data for CWE-362, which is 145 less than the old data (294 samples). The red dashed line indicates the P-value of 0.05. Any value above this line is insignificant.*

and inferences since the SVR model takes about four hours to train and test.

3.5 Discussion

This work not only covers the CVSS 3.1 scoring system but also the older CVSS 2.0 scoring system, which is the most dominant vulnerability evaluation metric in the NVD [24] [52]. Our proposed CVSS estimation method can convert the CVSS 2.0 score to the CVSS 3.1 score, and vice versa, solely from their natural language inputs. Compared to the works from [58] [86] [17], we showed the reports for each label in the component-wise task and a more end-to-end approach with the state-of-the-art performance. The USE

Table 3.8: *Resource Consumption (peak video memory usage in GBs; time in minutes; storage in GBs) for USE and SVM: the USE-V2 model trains & inferences faster than the USE-Large model but consumes more memory and storage. The SVM costs no VRAM to run and only requires 277 MB of main memory at peak usage. BERT model performs the best but with the highest VRAM consumption.*

Model	Memory (GB)	Time (minutes)	Storage (GB)	MAE
USE-V2	3.06	**10**	1.04	0.92
USE-Large	3.66	63	0.59	0.94
SVM	**0.27**	241	**0.02**	0.74
BERT	7.11	75	0.42	**0.72**

model outperforms the CVSS-BERT model in the component-wise task with better weighted F1 scores (except on availability component prediction). A comparative study between the deep learning models and the human domain experts shows that the deep learning models outperform the human experts in the CVSS estimation task by a large margin. The USE model also shows strong generalization ability on the recent data as shown in Section 3.4.4.

Poor descriptions, flawed CVSS processing, and low transparency in CVSS processing led to poor performance on the CVSS estimation by human experts [33] and AI agents. After MITRE enforces CNAs to use description templates, the prediction task becomes easier. The USE model performs better on the data collected 2022-2023 than the data collected before 2022. The experiments on the 2017 data split and the 2021 split show similar findings. It might be interesting to see how the model performs on the data in the future as the training turns obsolete. It is also interesting to examine how the model would perform with named entity-only inputs as one can extract only relevant components from the vulnerability description. As CVSS 4.0 rolls out, another

task would be to predict the CVSS 4.0 score from the vulnerability description. The conversion between CVSS 2.0, 3.1, and 4.0 scores would be another task to explore since NVD discontinued CVSS 2.0 after July 13th, 2022 [59].

The end-to-end trained AI model uses the natural language description of the vulnerability to predict the CVSS score, instead of the CVSS vector string. A good end-to-end model may predict all the CVSS vector string components correctly but still have a low CVSS score prediction accuracy due to a poor understanding of the CVSS formula.

For example, 'Buffer overflow in OpenBSD ping.' is a vulnerability with a CVSS 2.0 base score of 2.1. Our trained AI model predicts a severity score of 7.1. Is this critical error caused by the design flaw of the CVSS formula or is the model thinking that this vulnerability itself is bad? It's hard to tell. An AI model would not only need to understand the vulnerability impact and exploitability from the natural language input but also the CVSS formula itself on the arbitrary weighting. Even human experts struggle to understand or agree with the CVSS formula, as shown in the process of crafting the CVSS criteria. It is not surprising that the AI model would have a hard time having a strong predictive power on the CVSS score too. An AI could be accurate in predicting the CVSS vector string but not the CVSS score, because of the formula's unjustified design. Section 3.2.3 discusses this further, addressing human experts' performance in CVSS estimation.

The free-formed nature of the vulnerability description in the CVE listing and the NVD makes it hard for security analysts to understand the vulnerability and its impact efficiently. The lack of a 'ground truth' for the vulnerability description quality and the lack of a standard format for the vulnerability description make it hard to check the presence of the essential information in a vulnerability description. This observation motivates us to propose a solution that increases the efficiency for the security analysts to check the presence of the essential information in a vulnerability description. In Chapter 4, we call this solution 'essential aspect labeling' and we propose AI models to screen

vulnerability descriptions and label the presence of the essential aspects in a vulnerability description.

Chapter 4

Essential Aspect Labeling for Vulnerability Descriptions

As shown in Figure 4.1, according to the nvd.gist.gov website on the CVE process, the vulnerability management process begins with reports from assumed good-faith reporters. For each of these reports, vulnerability description is the first thing you see after opening a vulnerability entry. Other than the reference code for exploitation and demonstration, it is the only other source of information for anyone, especially the non-technical people, who wish to understand a flaw in some vulnerable software products. Reading the description of a vulnerability is the first step for any security analyst to understand the vulnerability and its impact. Since the vulnerability listing is largely a confederated work from various organizations and the open community with different levels of expertise, these free-form descriptions can often be incomplete, inconsistent, or even misinforming. For any vulnerability maintaining process, this information checking has been a time-consuming and labor-intensive task. This chapter proposes a solution that increases the efficiency for the security analysts to check the presence of the essential information in a vulnerability description.

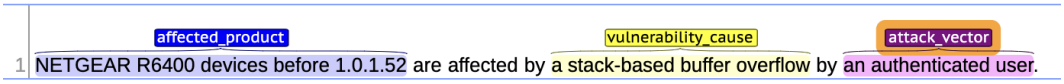
4.1 Problem Statement

Figure 4.1: *Vulnerability Management Process: The process begins with reports from assumed good-faith reporters. These reports are stored in a database with assigned metadata and subsequently analyzed by security analysts. Following the analysis, the vulnerabilities are assigned metrics that assist in their triage. End-users leverage these metrics to prioritize vulnerabilities for mitigation within their management systems.*

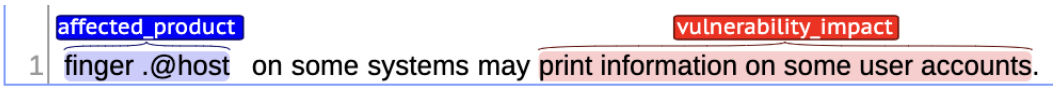
4.1.1 What is the issue with the current vulnerability description in the CVE listing and the NVD?

Daniel Stenberg, the author of cURL and the recipient of the 2017 Polhem Prize, has argued that the CVE system is 'full of questionable content and plenty of downright lies', 'next to impossible to get rid of invalid CVEs', bureaucratic and intransparent, and 'designed for good-faith reporters against bad-faith product organizations'. Anyone could submit a CVE entry with an anonymous account without any verification of the identity or the expertise of the reporter. Inevitably, there would be 'bad-faith reporters' who submit CVE entries with malicious intent, such as to gain fame , to cause damage to a system, or to gain financial benefit from the vulnerability disclosure.

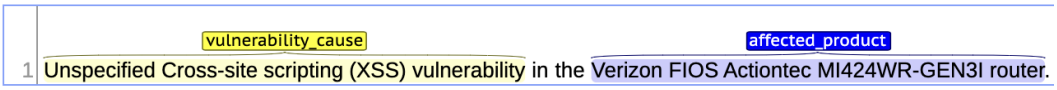
Even though NVD recommends reporters to follow two templates [15], CVE descriptions are still largely free-form and often lacking the essential information for a security analyst to understand the vulnerability cause and its impact. For instance, as shown in Figure 4.2, CVE-1999-0198 has the description 'finger .@host on some systems may print information on some user accounts.' This short description identifies the vulnerable product and potential impact of exploit with rather vague language such as 'some systems' and 'some user accounts.' It is unclear what kind of information the print-



(a) CVE-2021-38522 lacks the Vulnerability Impact aspect



(b) CVE-1999-0198 lacks the Attack Vector and the Vulnerability Cause aspect

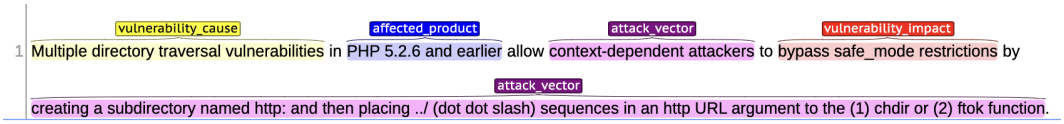


(c) CVE-2013-3097 lacks the attack vector aspect and the vulnerability impact aspect.

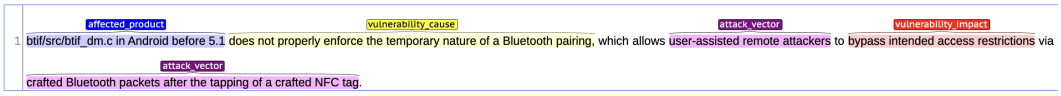
Figure 4.2: *CVE Descriptions Lacking One or More Essential Aspects*

out has and who can access it. This opaque description makes it hard for a security analyst to understand the vulnerability and its impact efficiently since ‘some‘ provides few insights on the vulnerable condition. The report should have included precise conditions that could trigger a successful exploit, such as the version of the vulnerable product, the environment, and the attacker’s privilege level, and the impact of the exploit as the potential damage to the system or the user. A better revision could be ‘finger .@host on Linux version before 2.2.14 may print information on user accounts with read privilege. This information may include the other user’s home directory, shell, and UID. An attacker with network access to the vulnerable system may use this information to gain unauthorized access to the system.’

As a first step to address the ‘bad-faith reporters‘, we propose syntax checking on the written descriptions for the four aspects of a vulnerability: the Affected Product, the Vulnerability Cause, the Attack Vector, and the Vulnerability Impact. If an entry lacks one or more of these aspects, it will be hard for a security analyst to fully understand the vulnerability and its impact since the reporter fails to provide the full context of the vulnerability. This



(a) CVE-2008-2666



(b) CVE-2014-7914

(c) CVE-2021-27402

Figure 4.3: *CVE Descriptions with All Four Essential Aspects*

checking would transfer the responsibility on proving the validity of a CVE from the affected CNAs to the more distributed reporters therefore release the CNAs from the burden of verifying the validity of the CVE entries.

However, our solution still has short-comings since it is possible for a reporter to submit a CVE entry with false semantics while syntactically being valid according to our checking model. Nontheless, syntactic highlighting could be helpful to spot the bugus CVEs before they roll into the world as 'disputed' CVEs, as no one could verify the authenticity of the report, and degrade the quality of the CVE database. With the rare case of 'good-faith reports' with carelessly written descriptions, our solution could nonetheless help the reporters to improve the quality of their descriptions.

4.1.2 What are the descriptions needed for a security analyst to understand the vulnerability?

We derive essential vulnerability aspects (EVAs) from the two CVE templates recommended by NVD [15] and works from Zhang et al. [88] and Guo et al. [30]. The fields identified by the two CVE templates have a fine-grained categorization of the vulnerability information but NVD gave these fields the

same level of importance without identifying which fields are essential for a security analyst to understand the vulnerability. In our proposed method, we identified the essential components for the CVE description and added a layer of abstraction to group similar fields into four essential aspects. In the VIET work from Zhang et al. [88], they used an NER LSTM model to extract entities at word-level granularity. Our proposed method uses a transformer model that works at the sentence-level granularity, not only extracting the entities but also classifying the sentences into the four essential aspects. Guo et al. [30] proposed a method to detect aspects that were often missing from CVE descriptions and provided suggestions to fulfill the missing aspects with CNN or LSTM models. Our proposed method builds on their work by defining a more comprehensive set of essential aspects and using a transformer-based model for aspect labeling.

4.1.3 What is the issue with vulnerability aspect labeling?

The proposed vulnerability aspect labeling task is a challenging task due to the limited amount of labeled data and the subjectivity of the labeling process. The small dataset could lead to overfitting for small-scale learning models and would degrade the performance of the model when compared to fine-tuned large-scale learning models [10]. With few-shot learning, even though not as performant as fine-tuned conterparts, large-scale language model might still outperform smaller models on this task. Increasing the size of the dataset, as a future and more engineering-oriented work, could help to improve the performance and the generalization ability of the model but finding the optimal size of any finetuning dataset requires unrealistic amount of experiments [87] and is therefore beyond the scope of this chapter.

The purpose of this chapter is to demonstrate the potential of the vulnerability aspect labeling task and to provide a baseline for future work. Therefore, we choose to use a small dataset and a few-shot learning approach to showcase

that low-cost AI deployment could provide at least some quality control over the CVE reporting process, instead of allowing free-form CVE descriptions to overload the CNAs with the responsibility of verifying the CVE entries. Any development team could take over from here to scale up the dataset size, the computation power, and the model size to improve the performance of the model and to provide a more accurate and reliable vulnerability aspect labeling service to the software vulnerability analysts.

4.1.4 Increase Model Size or Data Size?

Other than increasing the computation power, model size and data size are the two most common parameters to tweak for the performance of a machine learning model. Ideally, the more data the better, and the larger the model the better. According to Vieira et al., 100K+ data samples would be optimal for a LLM fine-tuning while data amount smaller than 2,000 would degrade the performance of the model [83]. Experiments from Google DeepMind also show that scaling up an LLM would benefit the learning task more than scaling up the dataset [87]. Larger models are also sample efficient [38], meaning that they could learn better from fewer data samples than smaller models. Therefore, this research focuses on model scaling and few-shot learning.

4.1.5 Research Scope

This research aims to analyze the Essential Vulnerability Aspects (EVAs) in the CVE description in real-time with few-shot learning LLMs. CVE description as the primary source of information for its analysts are often free-form and lack quality control. The proposed EVAs are the necessary types of information for CVSS scoring and CWE classification. The reporters and the analysts could have more highlighted description syntax on the CVE descriptions than the current free-form CVE descriptions. The goals of the EVA labeling are to remind the CVE reporters on what could be lacking within their CVE descriptions, to assist security analysts making informed and rele-

vant decisions on the CVE metrics, and to control the quality of CVE reports for the vulnerability database.

Since cyber defense systems rely on time critical data from its vulnerability database, real-time online analysis of the CVE data stream would be of great interest to the cyber defense community. With the evolved computation power and the development of LLMs, real-time online analysis of the CVE data stream is now possible due to the in-context learning capability of LLMs [10].

4.2 Related Works

This section reviews the related works in the field of vulnerability description quality assessment. We explain the previous work, their limitations, and how we build on them.

4.2.1 Defining Description Quality Metrics

The CVE program recommends that participating CNAs follow one of their two variations of templates when writing the descriptions for the vulnerabilities. These templates identify what fields a 'good' vulnerability description should contain, such as *vulnerability type, root cause, component, product, version, attacker, impact,* and *attack vector*. However, in the absence of complete information, it asks CNAs to provide only the *product*, *attacker*, *impact*, and *attack vector* fields hinting that the other fields are optional. [15].

The CNAs still write CVE descriptions in a freestyle manner without following the templates. This has led to a huge amount of variation in CVE descriptions where the CNAs could report the vulnerability with or without the necessary information. In an under-reported CVE, the CNAs would miss key details such as vulnerability impact or attack vector while in an over-reported CVE, they may have the details that do not contribute to the understanding of the vulnerability, such as enumerating product models and version numbers, making the majority of the report body [30] [79].

Kuehn et. al. evaluated the quality of descriptions by using three metrics, accuracy, completeness, and uniqueness [45]. To determine the description accuracy, they used a BERT model concatenated with the Flair NLP framework [3] to create the embedding for a classifier model. The uniqueness metric compares the number of fields required by CVE evaluation metrics such as CVSS within two CVE descriptions. It measures one description as more 'unique' than the other if it contains more fields required by target metrics than the other, thereby contributing more information to the database. However, this metric is more of an indicator of the quality of the whole database rather than of a single CVE description. The completeness metric uses a named-entity recognition (NER) model to extract the word or token from the descriptions, attempting to find the subject or the object that helps to determine the target metrics such as CVSS. However, these extracted tokens are heavily biased toward information related to attack vectors and vulnerability impact. The scheme also has double-counting issues. For instance, the descriptions with vulnerable functions or path fields not only count towards vulnerability cause but also towards attack vectors. The metric scheme would also count the vulnerability impact field three times as Confidentiality, Integrity, and Availability are highly correlated. Therefore, the completeness metric needs a weighting mechanism to balance the importance of the fields. The metric also ignores long Vulnerability Cause descriptions if there is no simple CWE wording for them due to the limitations of NER models. For complex metrics like Attack Complexity, the word-level NER model cannot provide meaningful discourse units for the analyst to make a decision. Therefore, we need a more powerful model that can predict at the sentence/sub-sentence level to correlate with the sophisticated fields.

4.2.2 Essential Aspects of Vulnerability Description

NER models are effective when dealing with a single token with clear boundaries. For tasks that require more context, models that work with text spans

work better. Span-level labeling can predict potentially overlapping text fragments with various lengths [22]. To analyse longer text spans, we adapted the term 'Vulnerability Aspect' from the work of Guo et al. [30] A vulnerability aspect is a coherent unit of information to fill a field in the CVE templates such as 'Vulnerability Type', 'Root Cause', 'Affected Product', 'Impact', 'Attacker Type', and 'Attack Vector'. In a CVE description, an aspect can span multiple words or phrases as long as it is a coherent unit of information to fill a field in the CVE templates.

Zhang et al. proposed an 'essential information' extraction method called VIET [88]. They defined essential information as named entities that can loosely map to the CVSS metrics. According to [88], a vulnerability should at least contain all the CVE fields that the CVSS metrics require to calculate the severity level. In Guo et. al's work, either the Vulnerability Type or the Root Cause aspects are necessary to the description, but not both, as Root Cause is often the Vulnerability Type described in simple language. For vulnerability without a popular type, one can only describe its Root Cause aspect. Therefore we combine Guo's Vulnerability Type aspect and Root Cause aspect into one 'Vulnerability Cause' aspect as detailed in Section 4.3.1. We also combine the Attacker Type with the Attack Vector in Guo's definition to form the new Attack Vector essential aspect due to ambiguity posed by the CVSS 'Attack Vector' metric [24], which is better mapped to the Attacker Type aspect, instead of the Attack Vector aspect [30], which would be more useful in determining the Attack Complexity metric in CVSS. The aspects as defined in Guo et. al. are inappropriate to measure description quality because the Attack Vector aspect fails to map to the CVSS Attack Vector metric, violating the 'essential property' defined by Zhang [88]. Therefore, we grouped Guo's aspects into four categories to better classify vulnerability descriptions: the 'Vulnerability Cause', the 'Affected Product', the 'Attack Vector', and the 'Vulnerability Impact', detailed in Table 4.1 as the 'Essential Aspects'.

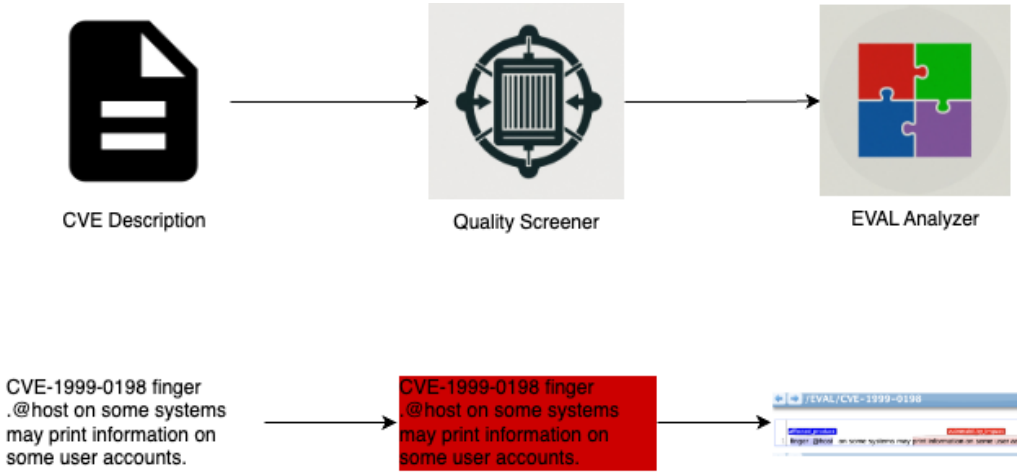


Figure 4.4: *Proposed EVAL Processing Pipeline for CVE Descriptions: the pipeline consists of two neural network models: the DesQS and the EVAL Analyzer. Firstly, a binary classification model DesQS screens for the poor descriptions within the data stream. Then another BART EVAL-Analyzer model identifies the missing essential aspects in the poor descriptions, by highlighting existing aspects. For instance, Quality Screener identifies CVE-1999-0198 as poorly described since it lacks the Vulnerability Cause and the Attack Vector aspects.*

4.3 Proposed Method

The proposed CVE processing pipeline consists of two neural network models: the Description Quality Screener (DesQS) and the Essential Vulnerability Aspect Labelling Analyzer (EVAL-Analyzer) as shown in Figure 4.4. The need for two models is due to the trade-off between classification accuracy and running cost. Pretrained LLMs such as BERT and BART require less data to train compared models trained from scratch. In the finetuning stage proposed in this chapter, for large models such BART-Large, they were trained with fewer epochs than smaller models such as BART-Base. Time is the determinating factor in considering the amount of training epochs involved in the training stage. This section details the Essential Vulnerability Aspect Labeling (EVAL) pipeline by explaining how we created the dataset in Section 4.3.1,

Table 4.1: *Relationship Between the Essential Aspects and the CVE Template Fields: the Essential Vulnerability Aspect (EVA) groups the CVE template aspects [15] into four supersets (V.C., A.P., A.V., and V.I.).*

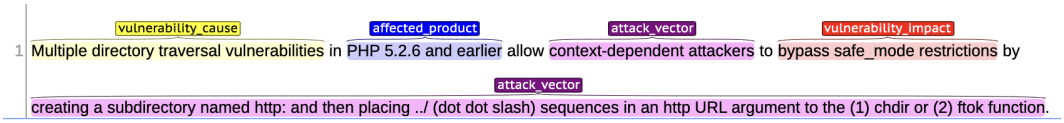
Essential Aspect	Template Fields
Vulnerability Cause	vulnerability type, root cause
Affected Product	component, vendor, product, version
Attack Vector	attacker, vector
Vulnerability Impact	impact

how we trained and evaluated the models for the DesQS in Section 4.3.2, and the EVAL Analyzer Section in 4.3.4.

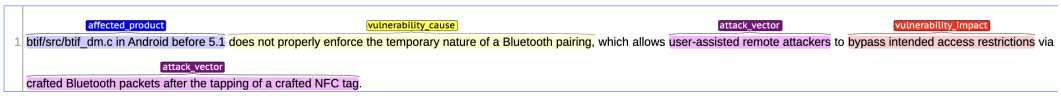
4.3.1 Essential Aspect Dataset

The Essential Vulnerability Aspect Labeling Dataset (EVAL Dataset) lays the foundation of the proposed EVAL pipeline as both the screening model and the analyzer model require the EVA-labeled data as shown in Table 4.1 to train. The dataset consists of 450 CVE descriptions with their essential aspects mannually labeled using the Brat annotation tool [78]. The Vulnerability Cause aspect combines the vulnerability type and the root cause fields in the CVE templates. The Affected Product aspect contains the component, vendor, product, and version fields, and provides the information of the vulnerable system. The Attack Vector aspect contains the attacker and vector fields, elaborating the situational context from the perspective of the attacker, while the Vulnerability Impact aspect, which describes the consequences of the vulnerability exploitation, and the change in the C.I.A. (Confidentiality, Integrity, and Availability) triad metrics, is the impact field in the CVE templates.

For the 450 entries in the EVAL dataset, the Datasets split function, implemented by Huggingface, randomly selects 405 entries for training and 45 for testing. However, we set the random state to 42 to control the computational

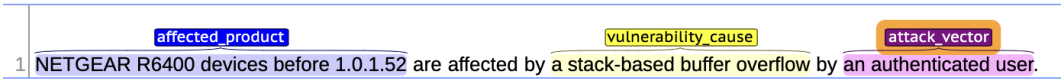


(a) CVE-2008-2666

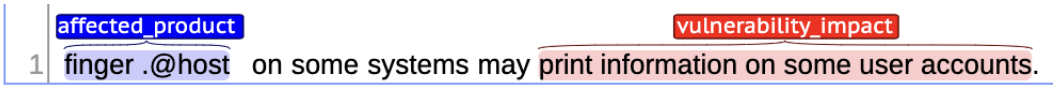


(b) CVE-2014-7914

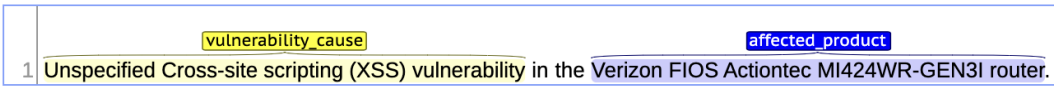
(c) CVE-2021-27402



(d) CVE-2021-38522



(e) CVE-1999-0198



(f) CVE-2013-3097

Figure 4.5: *Essential Vulnerability Aspect Labelling Dataset Examples in the Brat Labelling Tool: the first three CVE descriptions contain all four essential aspects with the Vulnerability Cause aspect positioned at the front (CVE-2008-2666), the middle (CVE-2014-7914), and the back (CVE-2021-27402). The last three descriptions (CVE-2021-38522, CVE-1999-0198, CVE-2013-3097) lack one or more of the essential aspects.*

environment to benchmark the results across different models and processing methods presented later in this section.

4.3.2 Vulnerability Description Quality Screener

We define a poorly written vulnerability description as a description that lacks one or more of the essential aspects. A script checks the aspect labels in each sample of the EVAL dataset. If an entry lacks one or more essential aspects, we define that entry as a poorly written vulnerability description, labeled as the integer 0. Otherwise, we label the entry with integer 1, indicating a well-written description that contains all four types of essential aspect labels. After this we randomly select one-third of the entries as the test data and the other two-third as the training data.

After training a BERT-based model on this dataset, we call the resulting model the Description Quality Screener (DesQS) model. The per device train batch size is 9, with a learning rate of 2e-5. The rest of the hyperparameters follow the default settings in the Huggingface Trainer class. For CWE-wise comparison, the training epochs are set to 27. For the severity-wise and vendor-wise comparisons, the training epochs are set to 36. These settings are arbitrary but kept consistent across all experiments for fair comparison. For each experiment, the model is retrained from scratch with the same hyperparameters and the same dataset split.

4.3.3 DesQS Model Performance

Table 4.2 shows that the DesQS model achieves a weighted F1 score of 0.755 with standard deviation of 0.032 on the 90 balanced test samples. The results are averaged over 30 runs with the same hyperparameters and dataset split. The tokenizer is en-uncased-preprocess version 3 from tensorflow-hub. The model is bert-en-uncased-l-10-h-128-a-2/versions 2. For each run, the model trains for 27 epochs.

Table 4.2: *Description Quality Result: BERT Model (Mean ± Standard Deviation)*

	Precision	**Recall**	**F1**	**Support**
Poor	0.773 ± 0.059	0.740 ± 0.059	0.752 ± 0.023	45
Good	0.751 ± 0.028	0.773 ± 0.098	0.757 ± 0.050	45
Accuracy			0.756 ± 0.030	90
Macro	0.762 ± 0.027	0.756 ± 0.030	0.755 ± 0.032	90
Weighted	0.762 ± 0.027	0.756 ± 0.030	0.755 ± 0.032	90

4.3.4 Essential Vulnerability Aspect Labeling Analyzer

The Essential Vulnerability Aspect Labeling Analyzer (EVAL Analyzer) not only segments the CVE descriptions into different aspects but also predicts the aspect types based on the surrounding context. Its main task is to identify the missing EVAs in the poorly written vulnerability descriptions to a fine-grained level as shown in Figure 4.5. We trained and compared three large language models, BART, T5 [68], and LED [8] each trained with different hyperparameters due to differences in their architecture. We set the batch size to 9 for all models if possible and to 1 for larger models like BART-Large due to limited GPU memory. The rest of the hyperparameters follow the default settings in the Huggingface Trainer class with the learning rate set to 2e-5. We train most models with epochs depending on the model architecture and the dataset size. For base models, the training epochs are set to 81; while for larger models, the training epochs are set to 18. For inference, we set the beam search size to 9, with top-k sampling of 18, temperature of 0.9, minimal length of 9, maximal length of 512, minimum new tokens of 9, and no repeat n-gram size of 3 to reduce text repetition.

The following are the evaluation metrics for the EVAL task.

Span Selection Recall counts how many spans in the reference can be found in

Table 4.3: *EVAL Analyzer Model Comparison: Mean ± Standard Deviation*

Model	Recall	Precision	**F1**	**Label %**	**Score**
BART-Base	0.882 ± 0.007	0.858 ± 0.009	0.865 ± 0.008	72.1 ± 1.49	62.4 ± 0.01
BART-Large	0.899 ± 0.009	0.887 ± 0.008	0.889 ± 0.007	68.1 ± 2.21	60.5 ± 0.02

the predicted text spans. '1.0' means all spans in the reference are found in the predicted spans.

Span Selection Precision counts how many spans in the predicted text can be found in the reference spans. Similar to the recall, '1.0' means all spans in the predicted text are found in the reference spans.

$$\text{F1} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{4.1}$$

Span Selection F1 is the harmonic mean of the recall and the precision as shown in the Equation 4.1.

Span Label Accuracy is the percentage of the predicted label types that match the reference. In order to find such a matching, first, a mapping script matches the predicted text spans to a target reference span. After the mapping step, another script counts how many labels are correctly predicted. To cope with the typos generated by the model, a semantic matching script uses the USE model to match the predicted labels to the reference labels with a cosine similarity bigger than 0.9.

Overall Score is the product of the label accuracy and the span selection F1 score, which is the final evaluation metric for the EVAL Analyzer models.

As shown in Table 4.3, the BART-base model has the best overall performance in the EVAL analysis task. To choose the best model for the EVAL task, we consider the F1 score, label accuracy, and the score after 9 runs with the same hyperparameters and dataset split. All the BART base models run for 81 epochs with a batch size of 9 with a semantic matching post-processing threshold of 0.9. The semantic matching uses the Universal Sentence Encoder

(USE) model [12] to calculate the cosine similarity between the predicted labels and the reference labels. The model is universal-sentence-encoder-large version 5 from tensorflow-hub. The BART-Large models run for 18 epochs with a batch size of 1 due to GPU memory limitations, with the same semantic matching threshold of 0.9.

The amount of training epochs is determined by the model architecture and the dataset size. To reflect real-world scenarios, we train the larger models with fewer epochs than the smaller models due to the time cost of training larger models. To fix training epochs for each model would impose unfair advantages depending on the model architecture, which is not ideal for a fair comparison. For smaller models such as BART-Base, we can afford to train for more epochs to achieve better performance; while for larger models such as BART-Large, we need to limit the training epochs to reduce the time cost and hardware requirements. If the bigger models are trained with the same epochs as the smaller models, they would have much better performance than the smaller models.

4.3.5 In Context Learning

As outlined in Table 4.4, the prompts used for the ICL tasks consist of two main components: **instruction** and **context**.

Instruction

The instruction conveys meta-information about the EVAL task. It specifically directs the model to output four essential aspects of the given input, formatted according to the Brat standoff annotation standard. This instruction remains identical across all ICL settings, ensuring consistency in the task requirements.

ICL	**Context**
Instruction	The Essential Vulnerability Aspect Labelling (EVAL) task highlights the Vulnerability Cause (VC), Vulnerability Impact (VI), Attack Vector (AV), and Affected Product (AP) information in the cve descriptions. Brat is a labelling tool. Please label the AP, VC, VI, and AV in the following cve description following the standoff format in Brat (text index, label type, and text span indices, labeled text).
Zero-shot	N/A.
One-shot	For example, the input CVE description: (demonstration.description), has the following output: '(demonstration.annotation)'
Few-shot	9 demonstrations+'What would be ideal annotation output for the following description: '
Many-shot	315 demonstrations+'What would be ideal annotation output for the following description: '

Table 4.4: *The Input Prompts Used for the In-Context Learning Task.*

Context

The context varies across different ICL configurations, primarily based on the number of examples provided:

- **Zero-shot learning:** No examples are provided to the model. It directly predicts the essential aspects of a given CVE description based solely on the instruction.
- **One-shot learning:** A single example is included in the context to guide the model.
- **Few-shot learning:** The context contains nine examples, offering the model more reference material for prediction.
- **Many-shot learning:** The context is enriched with 315 demonstrations, significantly increasing the number of examples provided.

Additionally, an experimental variation of the many-shot learning setup involves repeating the 315 demonstrations three times. This repetition aims to evaluate whether increasing the number of demonstration epochs enhances the model's ability to learn and generalize.

At the end of the context, a question is posed to prompt the model to generate the ideal EVAL annotation output for the given CVE description. By analyzing the outputs produced under these varying conditions, the study assesses the model's performance across different ICL configurations.

Data

The data used for the ICL tasks is sourced from the EVAL dataset, which contains 450 CVE descriptions with their corresponding essential aspects labeled. The dataset is split into demonstration set and testing, which is one-third of all the EVAL dataset. The 315 demonstrations are sequentially selected from the EVAL dataset. One-shot learning uses the first demonstration, while few-shot

learning uses the first nine demonstrations. For many-shot learning, the model learns all the demonstrations in the order they appear in the EVAL dataset.

Model

Gemini 1.5 Flash is the model of choice for the ICL experiments due to its cost effectiveness. The EVAL task proses a serious challenge to the Gemini 1.5 Flash model, as it requires the model to understand and segment the input and generate the essential aspects of the CVE descriptions while outputing the Brat standoff format. To accomplish this, we use Gemini's API to the GenerativeModel class provided by a Python library [28]. For the generation configuration, we set the temperature to 1 to encourage more deterministic outputs. The top P is set to 1 to allow the model to consider a wider range of possible outputs. The top K is set to 9 to limit the number of highest probability vocabulary tokens to consider during generation. The maximum output tokens are set to 8192 to accommodate the potentially lengthy Brat standoff format outputs.

4.3.6 Evaluation

The output of the model generation must meet the following criteria:

- **Brat Standoff Format** the output must adhere to the Brat standoff format, which includes the span index, label type, text span indices, and segmented text,split by a tab character. The indices output are ignored since a postscript can search for the correct indices conveniently.

- **Text Segmentation** the predicted spans must be related to one of the reference spans. A fuzzy matching alogrithm of edit distance maps each of the predicted spans to the reference spans.

- **Vulnerability Aspect Labels** the predicted label types for each span must match the reference spans' labels.

4.3.7 Post-Processing

Post-processing has two main components:

- **Find Brat Standoff** Predictions often include code segments in Markdown format, within which the Brat standoff annotations begin. To address this, a post-processing script is required to identify the start of the Brat standoff format, denoted by triple backticks in the Markdown structure.

- **Label Conversion** The model frequently predicts label types in abbreviated form, differing from the reference labels. For instance, the model may predict AV, while the reference label is attack_vector. A conversion script is therefore necessary to transcribe the predicted labels to align with the reference labels.

4.4 Result

In this section we attempt to answer the questions raised in Section 4.1. In the rest of this section we present the results of the experiments following the methods shown in the previous Section 3.3. All the experiments presented in this section are run on a single NVIDIA Tesla P100 GPU in Kaggle. The metrics with exact numbers are the best effort from the experiments.

4.4.1 In-Context Learning

Figure 4.6 shows the EVAL score results for the ICL tasks. Few-shot and many-shot learning achieve comparable EVAL scores, consistently surpassing zero-shot and one-shot settings. Repetitive demonstrations degrade performance, suggesting that the quality of demonstrations is more critical than their quantity. Figure 4.7 presents the span selection results across different ICL configurations. Both few-shot and many-shot learning consistently outperform zero-shot and one-shot settings in span selection recall, precision, and

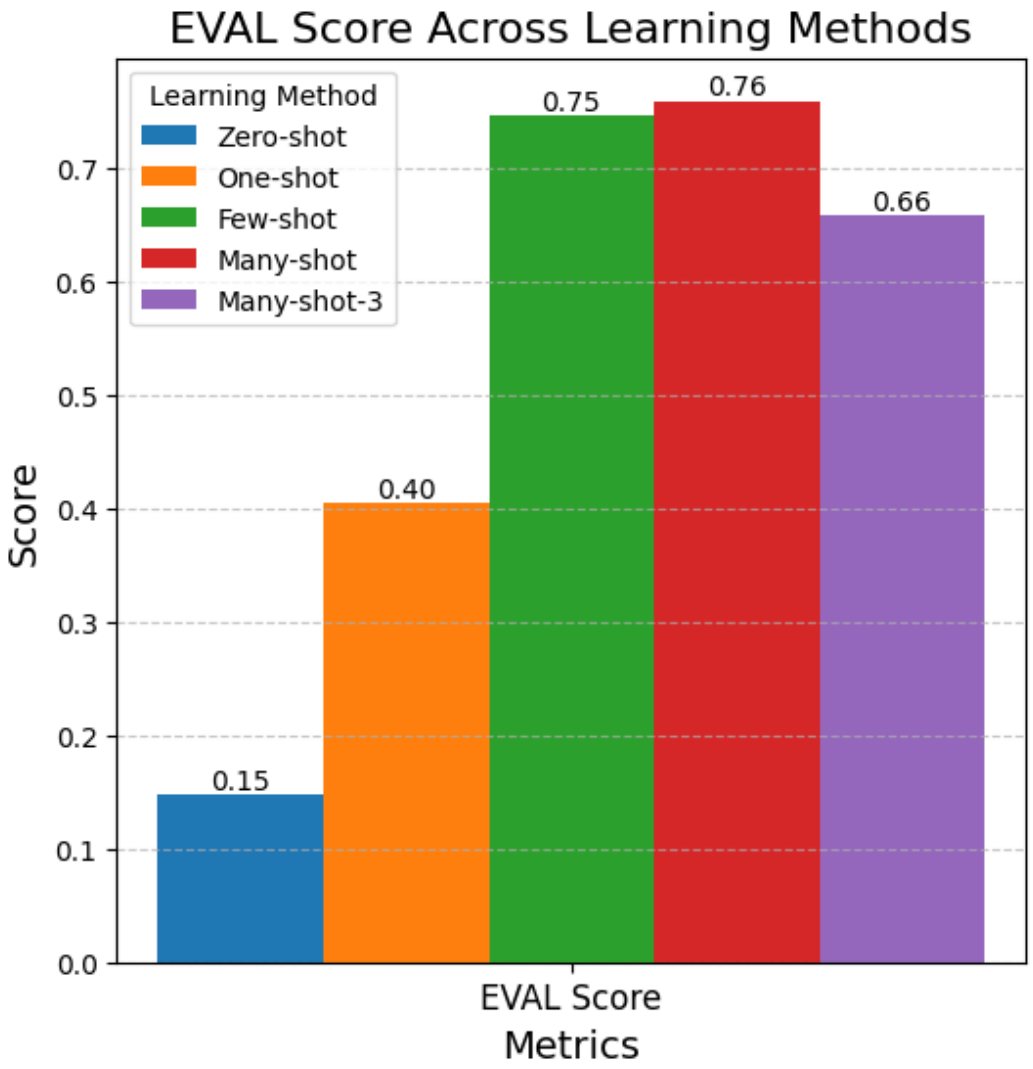


Figure 4.6: *EVAL Score Results: the many-shot and few-shot learnings consistently outperforms the zero-shot and one-shot learnings. More epochs of the same demonstration data would harm the model performance.*

F1 score. Figure 4.8 shows the label accuracy results for different ICL settings. For each span, we compare if the predicted label span is the same as the reference label span. Similar to the EVAL score results, few-shot and many-shot learning consistently outperform zero-shot and one-shot settings. Data diversity is crucial for model performance, as repetitive demonstrations lead to a decline in label accuracy. We also measured the amount of predictions that are perfectly aligned with the reference. By perfect, we mean exactly the same as the reference spans. Figure 4.9 shows few-shot learning has the best

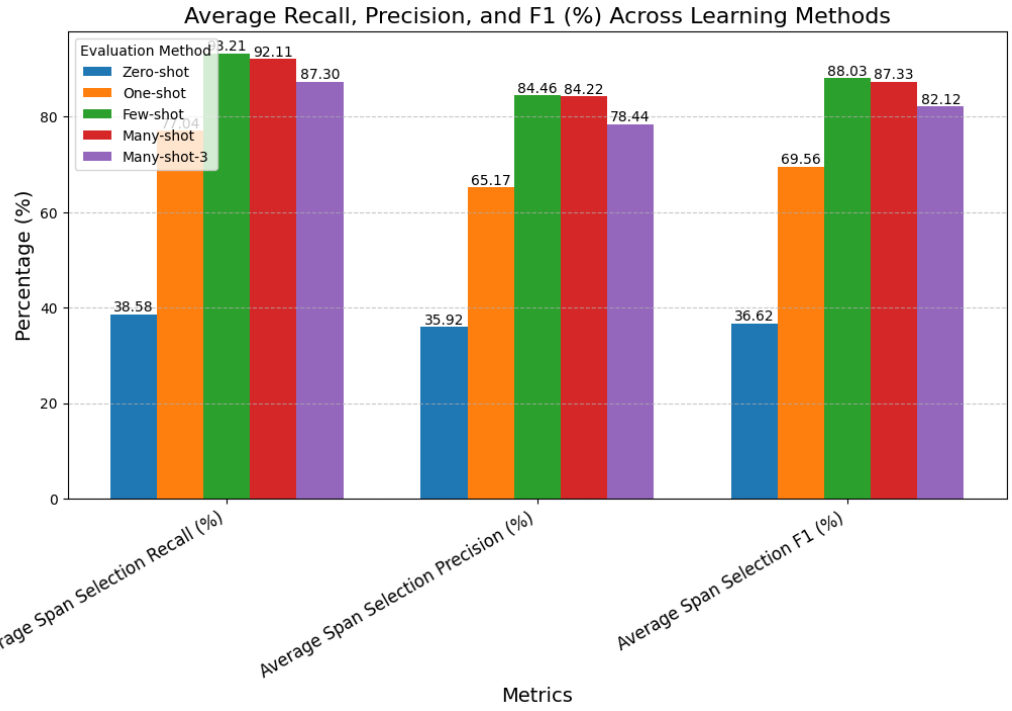


Figure 4.7: *Span Selection Results: few-shot learning and many-shot learning consistently outperform the zero-shot and one-shot learnings.*

performance when it comes to making the exact predictions with 28% of the predictions are perfect. Zero-shot learning never predicts an exact match with the reference. One-shot learning has merely 1% of the predictions are perfect. Many-shot learning achieves 17% while repeated demonstrations with three epochs decreased to 16%.

Overall, demonstrating with 9 examples could be the best strategy for the ICL EVAL task as shown in Figure 4.6, Figure 4.7, Figure 4.8, and Figure 4.9. This few-shot learning strategy consistently outperforms the zero-shot and one-shot learning strategies. After repeating the ICL and evaluation three more times, many-shot learning could perform better than the few-shot learning with respect to the EVAL score ranging from 0.70 to 0.76 or the label accuracy from 80 to 87 percent. Few-shot learning shows the best performance in terms of segmentation of the input description with the best average F1 score of 0.88 and predicting exactly the reference spans, achieving 28%. These results highlight the importance of data diversity and quality in improving the model

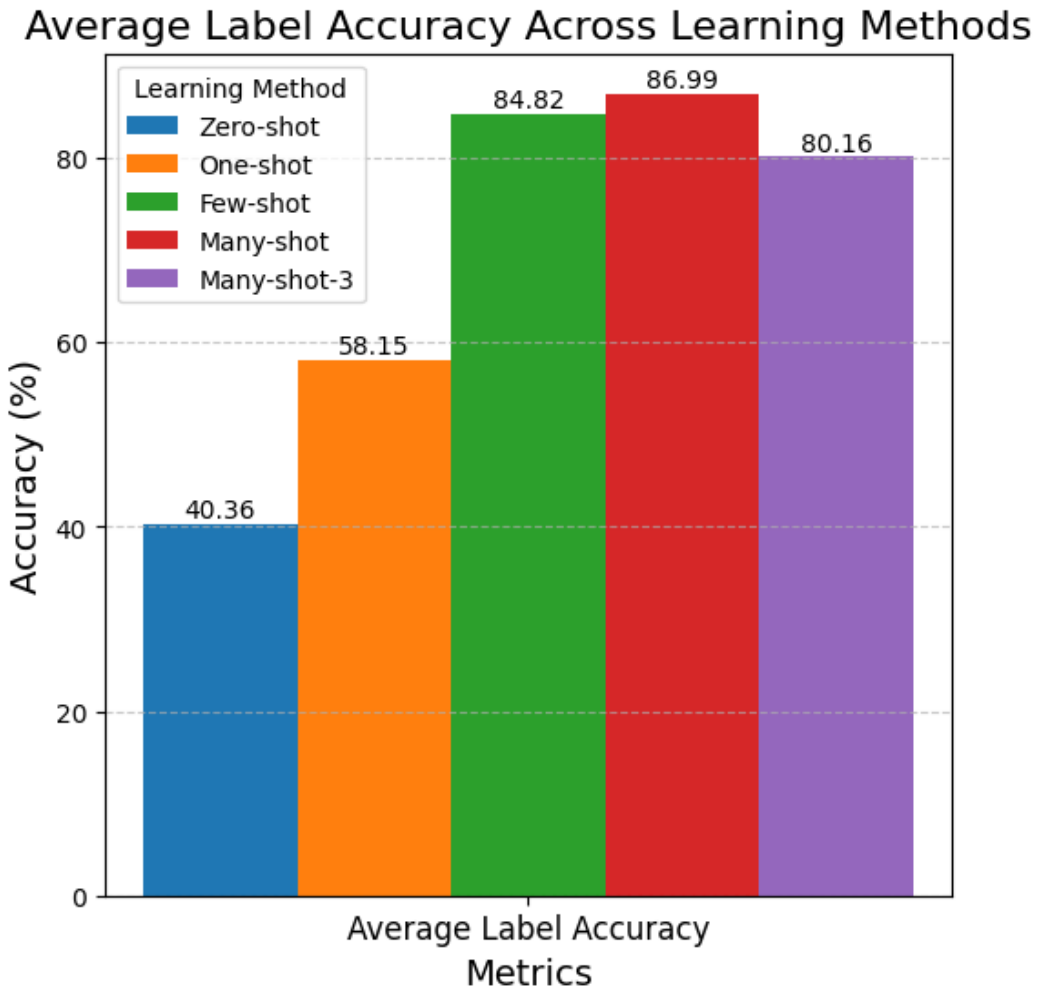


Figure 4.8: *Label Accuracy Results: few-shot learning and many-shot learning consistently outperform the zero-shot and one-shot learnings. More epochs of the same demonstration data would harm the model performance in label accuracy.*

for the EVAL as a in-context learning task.

4.4.2 Case Study on CVE-2024-0999

This section presents a case study on CVE-2024-0999, a vulnerability in the Totolink N200RE router, showcasing the effects of different learning settings on the EVAL model performance. CVE-2024-0999 has the following description:

A vulnerability was found in Totolink N200RE 9.3.5u.6139_B20201216.

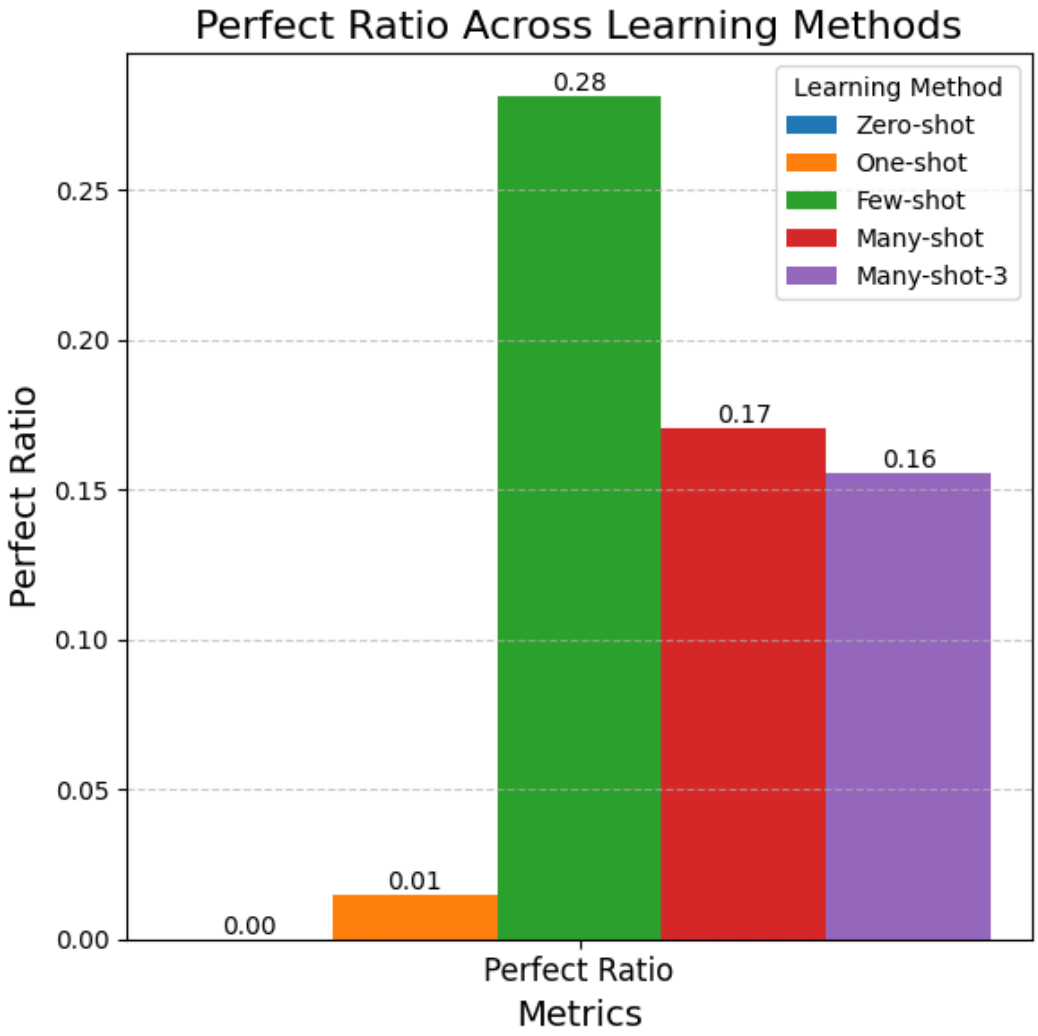


Figure 4.9: *Perfect Prediction Ratio Results: few-shot learning has the best result while zero-shot learning has the worst result. More demonstrations would cause the model to descrease in the amount of perfect predictions.*

It has been declared as critical.
This vulnerability affects the function setParentalRules
of the file /cgi-bin/cstecgi.cgi.
The manipulation of the argument eTime leads to
stack-based buffer overflow.
The attack can be initiated remotely.
The exploit has been disclosed to the public
and may be used.
The identifier of this vulnerability is VDB-252268.

```
NOTE: The vendor was contacted early
about this disclosure but did not respond in any way.
```

4.4.2.1 Zero-Shot Prediction

The following is the prompt used for the zero-shot prediction:

```
The Essential Vulnerability Aspect Labelling (EVAL) task
highlights the Vulnerability Cause (VC), Vulnerability
    Impact (VI),
Attack Vector (AV), and Affected Product (AP)
information in the CVE descriptions.
Brat is a labelling tool and visualized the
labelled output of EVALs.
Please label the AP, VC, VI, and AV in the
following CVE description
following the standoff format in Brat
(text index, label type, and
text span indices, labeled text):
A vulnerability was found
in Totolink N200RE 9.3.5u.6139_B20201216.
It has been declared as critical.
This vulnerability affects the function
setParentalRules of the file /cgi-bin/cstecgi.cgi.
The manipulation of the argument eTime
leads to stack-based buffer overflow.
The attack can be initiated remotely.
The exploit has been disclosed to the public and may be used
    .
The identifier of this vulnerability is VDB-252268.
NOTE: The vendor was contacted early about this disclosure
but did not respond in any way.
```

Gemini 1.5 Flash Model predicts the following aspects for CVE-2024-0999:

```
1 11 AP Totolink N200RE
11 12 AP 9.3.5u.6139_B20201216
28 45 VC setParentalRules
45 53 VC function
53 65 VC of the file
65 82 VC /cgi-bin/cstecgi.cgi
87 98 VC manipulation
98 104 VC of the
104 114 VC argument
114 122 VC eTime
122 137 VC leads to
137 162 VC stack-based buffer overflow
167 182 VI critical
187 202 AV remotely
202 211 AV The
211 219 AV attack
219 229 AV can be
229 236 AV initiated
236 244 AV remotely
249 269 AV The exploit has been disclosed to the
    public
269 285 AV and may be used
```

The predicted aspects are mostly correct other than spans that are broken into multiple parts. The BRAT standoff format is not followed in the prediction since T* span indications are missing.

BART-Large Model predicts the following in zero-shot ICL:

```
Please contact the vendor if you have any questions about
    this vulnerability.
```

The model fails to predict any of the essential aspects in the CVE description and instead provides a generic continuation to the 'Note' section of the CVE description.

4.4.2.2 One-Shot Prediction

The following is the prompt used for the one-shot prediction:

```
The Essential Vulnerability Aspect Labelling (EVAL) task
    highlights the Vulnerability Cause (VC), Vulnerability
    Impact (VI), Attack Vector (AV), and Affected Product (AP
    ) information in the CVE descriptions. Brat is a
    labelling tool and visualized the labelled output of
    EVALs. Please label the AP, VC, VI, and AV in the
    following CVE description following the standoff format
    in Brat (text index, label type, and text span indices,
    labeled text). For example, the input CVE description: '
    Jenkins Fortify Plugin 19.1.29 and earlier stores proxy
    server passwords unencrypted in job config.xml files on
    the Jenkins master where they can be viewed by users with
     Extended Read permission, or access to the master file
    system.' has the following output: 'T1
    affected_product 0 42   Jenkins Fortify Plugin 19.1.29
    and earlier T2   vulnerability_cause 43 130      stores
    proxy server passwords unencrypted in job config.xml
    files on the Jenkins master T3      vulnerability_impact
    149 164    viewed by users T4      attack_vector 170 231
       Extended Read permission, or access to the master file
     system' What's the standoff output for the input 'A
    vulnerability was found in Totolink N200RE 9.3.5u.6139
    _B20201216. It has been declared as critical. This
    vulnerability affects the function setParentalRules of
    the file /cgi-bin/cstecgi.cgi. The manipulation of the
```

argument eTime leads to stack-based buffer overflow. The attack can be initiated remotely. The exploit has been disclosed to the public and may be used. The identifier of this vulnerability is VDB-252268. NOTE: The vendor was contacted early about this disclosure but did not respond in any way.'?

Output for oneshot flash:

```
T1      affected_product 0 23   Totolink N200RE 9.3.5u.6139_B20201216
T2      vulnerability_cause 57 164      The manipulation of the argument eTime leads to
    stack-based buffer overflow
T3      vulnerability_impact 165 204    The attack can be initiated remotely
T4      attack_vector 205 247   The exploit has been disclosed to the public
```

The T3 is wrong since remote attacking should be classified as the attack_vector.

4.4.3 Impact of Essential Aspects on the Vulnerability Description Quality

A vulnerability description consists of multiple essential aspects but how much do each of the aspects impact the quality of a description? To answer this question, we first selected all 268 descriptions that had all four essential aspects from the total 450 labelled descriptions. From this dataset of 268 'good quality' descriptions, we selectively remove an essential aspect from half of them to craft the poor support samples. The other half of the 268 samples are kept as good supports. By doing so, we create a balanced dataset of 268 samples, with 134 poor quality samples (with one essential aspect removed) and 134 good quality samples (with all essential aspects present). After training a BERT model for 9 epochs with 178 training samples, we evaluate the model performance with 90 test samples, 45 poor test samples with an essential aspect eliminated for the description input, and 45 good samples.

Tables 4.6 4.7 4.8 4.9 show that after the four experiments with each removing an essential aspect, the DesQS model performs the best at predicting the Affected Product aspect, achieving perfect prediction accuracy (1.0 weighted F1), followed by the Vulnerability Impact aspect (0.97 weighted F1) while

Table 4.5: *Impact of Essential Aspects on the Description Quality*

Aspect	F1
Vulnerability Cause	0.87
Attack Vector	0.90
Vulnerability Impact	0.97
Affected Product	1.00

Table 4.6: *Result for Description Quality with Crafted Vulnerability Cause Examples*

	Precision	Recall	F1	Support
Poor	0.87	0.87	0.87	45
Good	0.87	0.87	0.87	45
Accuracy			0.87	90
Macro	0.87	0.87	0.87	90
Weighted	0.87	0.87	0.87	90

behaves the worst when the Vulnerability Cause aspect is absent from the description. In other words, missing the Vulnerability Cause aspect has the most impact on the description quality (0.87 weighted F1).

4.4.4 Relationship Between Description Quality and CVSS

Next, we wanted to understand if there was any correlation between the description quality and the CVSS 3.1 severity rating of vulnerabilities. In order to do this, we first categorized the CVEs into four groups according to their CVSS 3.1 Base Score: Low (0-3.9), Medium (4-6.9), High (7-8.9), and Critical (9-10). Then, we checked the quality of the descriptions in each group with DesQS model. Table 4.10 shows the result of the CVSS-3.1 Severity Level-Wise Description Quality Comparison. For each category, the table shows the number and percentage of good and poor descriptions. It can be seen that

Table 4.7: *Result for Description Quality with Crafted Attack Vector Examples*

	Precision	**Recall**	**F1**	**Support**
Poor	0.89	0.91	0.90	45
Good	0.91	0.89	0.90	45
Accuracy			0.90	90
Macro	0.90	0.90	0.90	90
Weighted	0.90	0.90	0.90	90

Table 4.8: *Result for Description Quality with Crafted Vulnerability Impact Examples*

	Precision	**Recall**	**F1**	**Support**
Poor	0.96	0.98	0.97	45
Good	0.98	0.96	0.97	45
Accuracy			0.97	90
Macro	0.97	0.97	0.97	90
Weighted	0.97	0.97	0.97	90

Table 4.9: *Result for Description Quality with Crafted Affected Product Examples*

	Precision	**Recall**	**F1**	**Support**
Poor	1.00	1.00	1.00	45
Good	1.00	1.00	1.00	45
Accuracy			1.00	90
Macro	1.00	1.00	1.00	90
Weighted	1.00	1.00	1.00	90

Table 4.10: *Description Quality Comparison: the numbers are the amount of CVE entries classified by the DesQ model. The 'Low' rating has the worst CVE description quality, while the 'Medium' rating has the best quality. The 'High' rating has the most data samples. MITRE Corporation has better CVE description quality than the CNAs, leading the comparison with a 25 difference in the percentage of good descriptions (67.5% vs 42.5%).*

Category	Size	**Good**	Poor
Low	1988	720(36.2%)	**1268(63.7%)**
Medium	41183	**20347(49.4%)**	20836(50.6%)
High	**44063**	20399(46.3%)	23664(53.7%)
Critical	15094	6519(43.2%)	8575(56.8%)
CNAs	94853	40349(42.5%)	54504(57.5%)
MITRE	107683	**72771(67.5%)**	34912(32.5%)

the 'Low' rating has the worst CVE description quality. The other three categories have an almost even distribution of good and poor descriptions with the 'Medium' category being marginally better than others.

The description quality increases slightly as the amount of data samples increases. However, this correlation between the number of samples and description quality is weak, as the medium severity rated data have 41183 samples, but the quality is the best among the four severity levels. Critically rated data have 15094 samples, but the quality is the second-worst among the four severity levels.

Table 4.10 clearly shows that low-severity vulnerabilities are more likely to have poor descriptions. However, does the perceived low severity cause people to write poor descriptions, or do poor vulnerability descriptions cause NVD to give them lower severity score. To investigate this relationship between the description quality and the CVSS score we used the same dataset of 268 samples with all the EVAs. We removed one EVA from each sample to craft

Table 4.11: *CVSS Score Change with Removed EVAs: removing the Vulnerability Impact aspect has the most significant impact on the CVSS score. Frquency is the percentage of samples that have a change in the CVSS score.*

Removed EVA	Change	Frequency
Vulnerability Cause	-0.026	0.619
Affected Product	0.172	0.615
Attack Vector	-0.201	0.668
Vulnerability Impact	**-0.446**	0.664

poor vulnerability descriptions. We then used the USE model from Chapter 3 to estimate the CVSS score of the poor samples. Table 4.11 shows the average change in the CVSS score after removing each EVA. Poor descriptions, especially ones without the Vulnerability Impact aspect, have significantly lower CVSS scores when compared with a good description, which leads us to conclude that vulnerability descriptions written poorly lead to lower CVSS score.

This we believe is an important result in light of the open source and collaborative nature of vulnerability disclosures, the resource constraints facing NVD as well as the importance of the CVSS scores that are relied upon by security teams around the world. A poorly written vulnerability description by a security researcher may lead to the underestimation of its severity, potentially resulting in security teams taking the vulnerability lightly.

4.4.5 Trends in Description Quality

Figure 4.10 shows the annual trend of CVE description quality measured using the DesQS model, from 1999 to 2022. The description quality steadily increased from 34.5% in 1999 to 85% in 2009. This behaviour is expected as MITRE analysts would have learned to write better descriptions with experience. However, after a few years of steady description quality of approximately

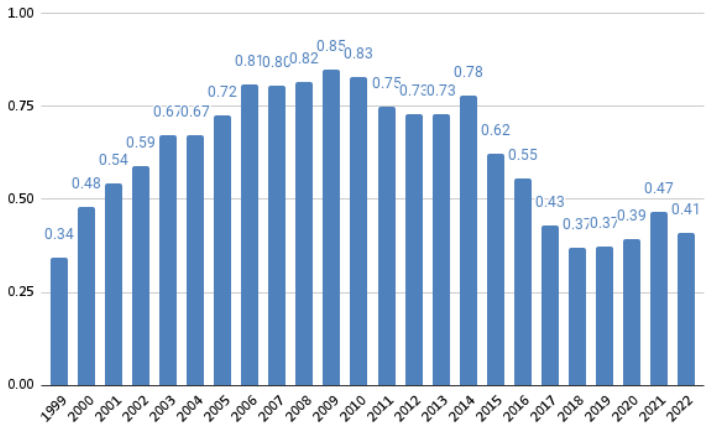


Figure 4.10: *Annual Trend of CVE Description Quality: the CVE description quality peaks in 2009 with 85% of the descriptions being well-written. In 1999, only 34.5% of the descriptions were good. In 2018, the quality touched another low at 36.9%.*

0.8, the quality tumbled to 43% in 2017 and 36.9% in 2018. Ever since 2018, the description quality has been fluctuating around 40%-50%. Notably, MITRE started to delegate the writing of CVE descriptions to CNAs in late 2016. This leads one to suspect that the CNAs are not writing as good CVE descriptions as MITRE, either deliberately or due to abuse of their policies, rather lacking the skills. The bottom part of table 4.10 shows the comparison between the MITRE and the CNAs and confirms the suspicion: MITRE has significantly better CVE description quality than the CNAs, leading the comparison with a 25 point difference in the percentage of good descriptions (67.5% vs 42.5%).

CVE database's increasing popularity within the security community has also brought it a significant issue. Along with genuine security researchers submitting CVE reports, CVE also sees a large number of 'credit-seekers' submitting sub-optimal reports that still need to be processed. One of the reasons behind the change to the federated model with CNAs was to enable the processing of the increasingly high number of CVE reports. The federated model allowed the number of published CVEs to more than double between 2016 and 2017. Our analysis, however, shows that the quality of CVE descriptions has

drastically gone down since the federated model started. It is arguable that poor quality vulnerability descriptions require more time for analysis, thereby leading to the clogging up of the NVD pipeline.

4.4.6 CNA-Wise Description Quality

It is important to identify CNAs that may have vulnerable internal policies leading to the acceptance and publication of low quality descriptions. To do this we first group the CVEs by their Assigner field in the NVD. Then, within each Assigner group, we screen the CNAs with the DesQS model, calculating the rate of good/poor descriptions over all the descriptions for each Assigner. Then, we rank the CNAs by their impact (positive impact minus negative impact) on the NVD.

In Table 4.12, we rank all of the 189 CNAs based on the description quality and show the resulting top 9 and bottom 9 CNAs. Good Submissions (Good) column is the amount of well-written descriptions submitted. Poor Submissions (Poor) column is the amount of poorly written descriptions submitted. Total Submissions (Total) column is the total amount of descriptions submitted. The rate of Good Submissions (Rate) column is the ratio of good descriptions over all submissions. The positive Impact (Positive) column is the ratio of the good submissions from this one particular CNA over all the good submissions from all the CNAs. The negative Impact (Negative) column is the ratio of the poor submissions from this one particular CNA over all the poor submissions from all the CNAs. The impact Rating (Impact) column is the Positive Impact minus the Negative Impact. Relative Score (Score) column is the ratio of the distance between the current CNA's Impact Rating and the highest Impact Rating over the distance between the lowest Impact Rating and the highest Impact Rating. According to our DesQS model results shown in Table 4.12, Oracle has the best overall impact on the NVD database, while IBM has the worst impact. TrendMicro has the best writing quality for CVEs, while SAP has the worst. The founding CNA, the MITRE Corporation, is

Table 4.12: *CNA Description Quality Ranking Top 9 and Bottom 9: Oracle has the best overall impact on the NVD database. TrendMicro has the best writing quality. The MITRE Corporation is a mediocre performer. However, since it has the most submissions, both good and poor, it makes it to the middle of the top 9. Qualcomm has the worst overall impact to the NVD database. IBM has the worst writing quality. SAP has the lowest percentage of good submissions.*

Assigner	Good	Poor	Rate	Positive	Negative	Impact	Score
Oracle	7044	1374	0.836	0.061	0.015	0.046	99.9
CERT	4148	777	0.842	0.036	0.009	0.027	78.5
Android	3221	616	0.839	0.028	0.007	0.021	71.3
Cisco/PSIRT	4301	2027	0.680	0.038	0.023	0.015	63.7
MITRE	**61112**	46066	0.570	**0.535**	**0.521**	0.014	62.5
TrendMicro	1099	**4**	**0.996**	0.010	0.000	0.010	57.8
Cisco/Talos	1164	80	0.936	0.010	0.001	0.009	57.5
Mozilla	1328	269	0.832	0.012	0.003	0.009	56.7
RedHat	6412	4376	0.594	0.056	0.050	0.006	54.4
Qualcomm	845	1092	0.436	0.007	0.012	-0.005	41.0
HPE	110	621	0.150	0.001	0.007	-0.006	39.7
SAP	**73**	627	**0.104**	0.001	0.007	-0.006	39.3
Huawei	359	853	0.296	0.003	0.010	-0.007	39.2
DHS	852	1695	0.335	0.007	0.019	-0.012	33.2
Apple	2211	3012	0.423	0.019	0.034	-0.015	29.7
Microsoft	**3341**	4113	0.448	0.029	0.047	-0.017	26.8
Adobe	1537	3117	0.330	0.013	0.035	-0.022	21.6
IBM	1101	**4429**	0.199	**0.010**	**0.050**	-0.040	0.0

right in the middle of the top 9.

4.4.7 Impact of Weakness Type on Vulnerability Description Quality

If different aspects have different impact on the description quality, then what about different weakness types? Essentially, we want to understand if certain weaknesses are harder to describe than others, as seen from CVE descriptions. In order to do this, from the 2023 Top 25 CWE List [27], we identify the group of CWEs with description ratings below 0.4 with the DesQS model.

Table 4.13 shows the description quality of the top 25 CWE entries in the NVD [27], according to a trained BERT model. The BERT model has an accuracy of 80% on the test set. The table clearly shows that there are weaknesses that have a much higher quality rating and others that have a much lower quality rating than average. CWE-89, CWE-22, and CWE-94 have description quality ratings above 0.7 while CWE-798, CWE-918, CWE-306 and CWE-269 have description quality ratings below 0.4. We further analysed the aspects that are missing in the CVE description for each of these CWEs using the EVAL Analyzer model.

4.4.7.1 EVAL Analysis

For each CWE in Table 4.14, we report the number of missing essential aspects in the CVE descriptions; the size of the test dataset; the DesQS Screener model reported ratio of poor quality descriptions; the EVAL Analyzer Analyzer model reported poor rate; and the difference between the two ratios. The last row shows the total number of missing essential aspects in the CVE descriptions within the four CWEs, the average poor rate of the two models and the difference, which elaborates how much the higher resolution EVAL Analyzer Analyzer model differs from the DesQS Screener model.

The results show that Vulnerability Impact is the most frequently missing aspect in the CVE descriptions of these CWEs, followed by the Attack Vector

Table 4.13: *Top 25 CWEs ([27]) Description Quality : CWE-79 'Cross-site Scripting' has the most entries in the NVD database while CWE-918 'Server-Side Request Forgery' has the least entries. CWE-89 'SQL Injection' has the highest rating among the top 25 CWEs while CWE-306 'Missing Authentication for Critical Function' has the lowest rating.*

Rank	ID	Size	Good	Poor	Rating
1	CWE-787	5345	3128	2217	0.585
2	**CWE-79**	**17760**	10440	7320	0.587
3	**CWE-89**	7098	5432	1666	**0.765**
4	CWE-416	2390	1442	948	0.603
5	CWE-78	1923	1086	837	0.564
6	CWE-20	8547	4778	3769	0.559
7	CWE-125	3835	2141	1694	0.558
8	CWE-22	4192	2990	1202	0.713
9	CWE-352	3414	1813	1601	0.531
10	CWE-434	997	455	542	0.456
11	CWE-862	753	413	340	0.548
12	CWE-476	1672	718	954	0.429
13	CWE-287	2589	1188	1401	0.458
14	CWE-190	1530	678	852	0.443
15	CWE-502	691	332	359	0.480
16	CWE-77	920	454	466	0.493
17	CWE-119	11463	7760	3703	0.676
18	CWE-798	737	252	485	0.341
19	**CWE-918**	**515**	174	341	0.337
20	**CWE-306**	577	189	388	**0.327**
21	CWE-362	968	514	454	0.531
22	CWE-269	1515	552	963	0.364
23	CWE-94	2733	2010	723	0.735
24	CWE-863	1034	463	571	0.447
25	CWE-276	547	300	247	0.548

Table 4.14: *Missing EVA Analysis: Vulnerability Impact (V.I.) is the most frequently missing aspect in the tested CWEs descriptions. The Screener model overestimates the quality of the poor CVE descriptions by an average of 19.1%.*

CWE/CNA	**V.I.**	V.C.	A.P.	A.V.	Size	Screener	Analyzer	Difference
CWE-798	346	50	8	221	737	0.659	0.526	-0.133
CWE-918	174	341	0	0	515	0.663	0.450	-0.213
CWE-306	189	78	4	388	577	0.673	0.386	-0.287
CWE-269	552	178	963	299	1515	0.636	0.497	-0.139
Total	1250	344	975	707	3934	0.658	0.467	-0.191
SAP	203	153	31	**236**	700	0.896	0.497	-0.399
IBM	**1364**	537	2	1243	5530	0.801	0.342	-0.459
Microsoft	**2444**	488	750	2402	7454	0.552	0.360	-0.192
Adobe	**1747**	879	0	1468	4654	0.67	0.559	-0.111
Total	**5758**	2057	783	4349	18338	0.729	0.439	-0.290

aspect. Affected Products is almost always present in the CVE descriptions. Occasionally, these CVE descriptions lack the Vulnerability Cause aspect. This is interesting because in general as shown in Section 4.4.3 Vulnerability Cause and not Vulnerability Impact affects the description quality the most.

The CWE-wise description quality ratings are different in Table 4.13 and Table 4.14 because Table 4.13 shows the overall description quality of the CWEs with a BERT based Screener model while Table 4.14, using the BART based Analyzer model, shows not only the quality but also the missing essential aspects in the CVE descriptions within the select CWEs.

The DesQS (Screener) model overestimates the quality of the poor CVE descriptions by 13.3% on CWE-798, 21.3% on CWE-918, 28.7% on CWE-306, and 13.9% on CWE-269.

Table 4.15: *Running Time Comparison: USE semantic matching imposes 2x running time increase. The base models could work with larger batch sizes and more epochs. The screener BERT model inferences much faster than the analyzer BART model. Note that screening data is much larger than the analysis data for the CWE-269 analysis, therefore the running time is not directly comparable.*

Task	Model	Batch Size	Epochs	Time(H)
CNA Screening	**BERT-Base-Cased**	9	36	1.12
CWE-269 Analysis	BART-Large-SM	1	18	8+
CWE-269 Analysis	**BART-Large-LD**	1	18	0.75
CWE-269 Analysis	BART-Base-SM	9	81	1.88
Microsoft Analysis	BART-Large-LD	1	18	3.36

4.4.7.2 CNA Missing EVA Analysis

Similar to the CWE-wise analysis, we also analyzed the missing essential aspects in the CVE descriptions within each CNA. The bottom part of Table 4.14 shows that the Vulnerability Impact aspect is the most frequently missing aspect in the tested CNAs CVE descriptions, with Attack Vector being the second most frequently missing aspect. The Screener model overestimates the quality of the poor CVE descriptions by 39.9% on SAP, 45.9% on IBM, 19.2% on Microsoft, and 11.1% on Adobe, resulting in an average of 29.0% overestimation.

4.4.8 Running Cost Comparison

The running time cost includes both the training and the evaluation inferencing. For the BERT model in Table 4.15, we sampled the most time-consuming screening task Vendor Description Quality Ranking of 189 CNAs as in Table 4.12. The training of the BERT Screener model cost 6 minutes and 43 seconds with a storage cost of 436MB for the safe tensors of the base model. The

inference time for the 189 CNAs took exactly 1.02 hours, costing a total of 4057.7 seconds.

For the EVAL Analyzer models, we presented the most time-consuming task done so far, CWE-269 EVAL analysis with BART-Large model, which costs 1.02GB storage space, with the USE model as semantic matching, which forced Kaggle to kill the kernel due to the long-running time of 8+ hours, of which the training of the BART-Large model took 42 minutes 41 seconds on the 405 hand-labeled samples. Under the semantic matching scheme, an Analyzer model would need to segment the CVE descriptions into the essential vulnerability aspects, and then label each of the EVAs. For each label of the EVA, another USE model encodes the reference label and the predicted aspect and then compares the cosine similarity between the two. Even though the USE similarity comparison increased labeling matching performance, it also increased the running time, considering CWE-269 has 1515 samples. Switching to a base BART model helped to reduce the running time to 1.88 hours as shown in Table 4.15 but the EVA segmentation accuracy started to suffer. We had to come up with a quicker solution for post-processing.

The Levenshtein Distance matching greatly reduces the overhead when doing experiments with large-scale sample evaluation datasets such as CNA vs MITRE comparison, CWE-Wise Description Quality, and Poor Description and Poor CVSS Score shown in Section 4.4.7. In the most significant case, Levenshtein Distance matching reduces the running time from 8+ hours to 0.75 hours while running experiments for the CWE-269 EVAL analysis, achieving comparable running time cost even with the larger scale screening task of the BERT-base model. However, the Screener model inference time is much faster than the Analyzer model inference time. For instance, the Analyzer model took 3.36 hours to analyze just the Microsoft CNA, while the Screener model took only 1.12 hours to screen all the 189 CNAs, which is exactly 3 times faster.

4.5 Limitations

Due to the academic setting of this research, this exploration on the potential applicability of LLMs in the cyberspace has limited research time, human power, and computational hardware. Therefore, production grade fine-tuning is infeasible due to the quantity of the required data samples and the amount of VRAMs needed for the LLM fine-tuning. Since the labeled data is manually labeled, it is prone to subjectivities, errors and inconsistencies. Due to the limited time and resources, we only explored models shown in the result section with limited runs for each experiment. However, by no means, these results are exausitive of all possible models. To achieve better training efficiency, increasing model size is a key factor to the task performance [38]. But the larger the model, the bigger the VRAM requirements are for it to load and train. Therefore, we only explored models within the reach of mainstream consumers hardwares at the time of writing this chapter. This research choice also reflects the real-world constraints within the cybersecurity industry, where the security budgets are often limited and resources are scarce.

The training of an efficient EVAL model should scale data, model size, and computational power in a balanced way [38]. As computer hardware evolves, revisiting the EVAL model with more computational power could improve the model performance and efficiency, as well as the scalability of the model to larger datasets. However, this is more suitable for development teams in the industry with sufficient resources.

Vulnerabilities are constantly being discovered and published. Reducing the time window between the vulnerability discovery and the vulnerability classification could improve the efficiency of the vulnerability response and mitigation. Therefore, an online EVAL model that can quality control the CVE reports would be vital to the defence of potential cyber attacks.

Chapter 5

Explainable Common Weakness Enumeration Mapping

'Root Cause Mapping' has been an open problem in the cybersecurity domain. For each CVE entry, cybersecurity analysts need to identify the root cause of the vulnerability and map it to a Common Weakness Enumeration (CWE) label. This is a challenging task since the CVE descriptions are often noisy and inconsistent, and the CWEs have an hierarchical structure. One CVE may also have multiple CWEs assigned to it, and the CWEs may have different interpretations by different organizations. An automated solution with explainable classification would be beneficial to the cybersecurity analysts to either kickstart their analysis or to validate their analysis. This chapter explores the potential of using Long Context Large Language Models to solve the 'root cause mapping' problem by classifying Common Weakness Enumeration labels for Common Vulnerabilities and Exposures based on their descriptions.

5.1 Problem Statement

This section describes the problem of CWE mapping and the challenges associated with it. First section identifies the challgenes of the CWE mapping problem. Second section describes the research question and objectives.

5.1.1 Challenges

This section describes the challenges of the CWE mapping problem.

5.1.1.1 Inconsistent Labels

Table 5.1: *CWE Mismatch*

CVE-ID	MITRE	NIST	CNA
CVE-2022-24730	CWE-862	CWE-863,CWE-22	CWE-284
CVE-2009-3597	CWE-862	CWE-552	N/A.
CVE-2008-5027	CWE-862	CWE-264	N/A.
CVE-2009-2960	CWE-862	CWE-264	N/A.
CVE-2007-2925	CWE-862	NVD-CWE-Other	N/A.
CVE-2006-6679	CWE-862	CWE-863	N/A.
CVE-2005-2801	CWE-264	CWE-697	N/A.
CVE-2001-1555	CWE-269	NVD-CWE-Other	N/A.
CVE-2001-1514	CWE-269	NVD-CWE-Other	N/A.
CVE-2001-0128	CWE-269	NVD-CWE-Other	N/A.
CVE-2007-4217	CWE-269	CWE-119	N/A.

MITRE Corp. elaborates each CWE with an ‘Observed Examples‘ section that are accessible through its website. However, these examples often have inconsistently labeled CWEs as NIST or CNAs could have different interpretations of a given vulnerability. For example, Table 5.1 shows the CVE examples that have conflicting classifications in CWE-862. MITRE Corp. has particularly mislabelled CWE-269, since CWE-269 is a discouraged mapping. In their enumerated examples for CWE-269, only CVE-2007-5159 has a conformed label from NVD, who treat the rest mostly as ‘NVD-CWE-Other’.

5.1.1.2 Labeling Complexity

Classifying CVEs can be difficult since there might be multiple CWEs assigned to a single CVE. Around 5 percent of the CVEs in the NVD have multiple CWEs assigned to them. But this number could be larger since some reporters follow bad practice to duplicate the CVEs with different CWEs, while they could merge the CVEs and assign multiple CWEs to a single CVE instead [61].

Also, CWEs have an hierarchical structure, which means that a parent CWE could have multiple child CWEs. One could map a CVE to a parent CWE, but it would be more informative to map it to the child CWEs. However both the parent and the child CWEs could be correct labels for a CVE. This makes the labeling of the CVEs and the evaluation of the label quality rather complex since there are technically multiple correct CWEs assignable to a CVE. Therefore, the evaluation of the CWE labeling should contain a measure of the CWE similarity.

5.1.2 Research Question and Objectives

How can a model achieve high tolerance to data noise?

The CVE descriptions are often noisy and inconsistent as discussed in Section 5.1.1, which could affect the performance of the classification models. Therefore, the model should be able to tolerate the noise in the data, which mostly consist of positive or unlabeled examples. Noisy data could prevent a model or even humans from making correct classification decisions.

What techniques enable fast online inference?

Since cybersecurity tasks are highly time-critical, the model should be able to make the classification decisions in a timely manner. Online learning could be a potential solution to this problem since the model could be optimized on the fly with a few examples and instructions. When time is limited, classifi-

cation of CWEs could be error-prone, making explainable classification more challenging.

How to provide explainable and highlighted CWE classification?

To help the security analysts to understand the classification decisions, the model should be able to provide explanations and highlights for the CWE labels. A highlight is a part of the CVE description input that supports the labeling decision. The CWE explanation should be able to provide the rationales for selecting the CWE label. The highlights should be the keywords.

5.2 Related Work

Researches on CVE classification or the 'root cause mapping' problem have been limited. This section only presents the most relevant work. As a challenging multilabel machine learning task, Oostveen identified major issues that could affect the performance of the CVE classification models [61]. Even though CVE classification is multilabel, the majority of the CVEs have only one CWE assigned to them and only 5 percent of the CVEs have multiple CWEs assigned to them. There are also plenty of human errors in the CVE listings. As Oostveen [61] pointed out that there was a case where NVD kept information from Microsoft secretive and consequentially the CVE description made by NVD lacks details in their database for this particular CVE. Oostveen generated a dataset by combining duplicate CVEs of different CWE labels. However, this dataset is still a rather challenging dataset to work with since there are human errors in the original NVD dataset due to disagreements among the CNAs and wrong labelling. The sheer number of CNAs of different levels of CWE expertise could also affect the quality of the dataset.

Table 5.2: *Example Data: input format: CVE := Description,output format: CWE := Explaination — Highlight*

Input	Output
CVE-2020-1432 := An information disclosure vulnerability exists when Skype for Business is accessed via Internet Explorer, aka 'Skype for Business via Internet Explorer Information Disclosure Vulnerability	NVD-CWE-noinfo := The description doesn't provide specific technical details; it states an ""information disclosure"" vulnerability. This implies that Skype for Business, when accessed through Internet Explorer, might leak sensitive data that should be kept confidential. This could range from user information to internal application details, potentially aiding attackers in further compromising the system or users. — information disclosure vulnerability

5.3 Methodology

This section describes the dataset, the model, and the inference process used for the CWE in-context classification task.

5.3.1 Dataset

The dataset is manually labeled and consists of 108 CVE descriptions with CWE labels. One-third of the dataset (36 entries) is used for testing and the rest is used for demonstrating. Demonstrations are examples that are used to show the model how to do the task. These examples are similar to training examples in fine-tuning settings, but they are not used to train the model. Instead, they injected into the model as part of the prompt.

The inputs are the CVE ID and CVE description. Even though the CVE IDs themselves are not useful in the classification task, they are used to identify the CVE descriptions and the corresponding CWEs. CVE references are links or resources that provide additional information about the CVE. While reference could also benefit the model for making the labeling decision, esspecially when the description follows a generic template, due to techinicality of crawling each link which potentially be credential and vendor-specific, reference is not included in the input.

The outputs are the CWE labels, explanations, and highlights. The CWE labels come from the NVD database and CNA reports. If there were any conflicts opinions, we would take the union of the contexted labels. This assumes that reporters have different opinions on the same CVE but are assumed to never make mistakes. While this assumption could be rather naive, the amount of work to differentiate the mistakes from the disagreements would be rather large. This could be a promising future work of research similar to how the CWE mapping guidelines suggest, the explanations on how to find the relavant CWE label would be of the interest of the security analysts. For each CVE and the label, the explanation or the rationales for selecting this label are provided. Highlights are part of the CVE description input that support the labeling decision. These highlights could be the keywords that are used to determine the CWE label.

The main goal of this dataset is to demonstrate the potential of the LLMs used in the CVE classification task, not to provide an industry-grade dataset that could be used in a production environment. The dataset is designed to work in the ICL scenario, where the model is provided with the annotated examples in-context, and then used to classify the CVE descriptions into CWEs with explanations and highlights. One could fine-tune with these examples but the effect might be limited since the dataset is rather small and therefore prone to overfitting.

5.3.2 Model and Inference

The model we used is the Llama 3 with gradient variant (gradientai/Llama-3-8B-Instruct-Gradient-1048k) with a text-generation pipeline from the Transformers library from hugging face with datatype set to bfloat16 loaded onto two NVIDIA RTX A6000 GPU, each equiped with 49GB of VRAM. To work with multiple GPUs, the Accelerate library is deployed [29]. For the generation parameters, the max new tokens is set to 256 to allow enough space for the model to generate the CWE label, explanation, and highlight. The temperature is set to 0.6 to allow some randomness in the generation. The top-p is set to 0.9 to allow the model to consider a wider range of possible tokens.

The task involves classifying CVE descriptions into CWEs with explanations. A task-specific prompt format was designed, including annotated examples (demonstrations) drawn from the training data to enable in-context learning. The prompt structure includes task instructions and demonstrations in the format CVE := Description mapping to CWE := Explanation — Highlight. The CVE description was appended to the prompt and passed to the LLM for prediction. Sampling parameters (e.g., temperature and top-p) were adjusted to balance diversity and relevance in the generated output. The predictions were parsed to extract CWEs using a regex-based post-processing step.

5.3.3 Rule Book

The Rule Book (RB) is a set of rules that the model should follow when making predictions. It is supplimented to the prompt to feed into the model. The rules are explicit knowledge that is injected into the model to help with the performance on the task. Rules enlisted in RB are generalized and are not specific to a particular CVE or CWE that directly give out the CWE label. To come up with these rules, the outputs of the model were analyzed and the patterns were extracted manually to boost the model performance with best effort.

5.3.4 Evaluation

A seperate evaluation Jupyter notebook on Kaggle loads a test dataset and commands a trained model to predict the target CWE label, explanation, and highlight. By comparing the reference CWE with the predicted CWE labels, the SciKit learn library [64] reports the classification statistics such as Precision, Recall, and F1. A labeled dataset containing CVE descriptions and corresponding CWEs was split into two-third for training and one-third for testing sets. The training data provided the annotated examples for the prompt, while the test data evaluated model performance. Predicted CWEs were extracted and compared to the ground-truth labels. Predictions, references, and the match ratio (percentage of correct predictions) were saved to a JSON file for analysis and reproducibility.

5.3.4.1 Group Matching for CWE Labels

The Group Match algorithm is designed to evaluate whether a predicted CWE label matches a reference CWE label, either directly or by association with the same group.

The Group Match algorithm determines equivalence between predicted and reference CWE labels based on:

- **Direct Match**: The predicted CWE is identical to the reference CWE.
- **Group Match**: Both CWEs belong to a predefined CWE group sharing common characteristics or relationships.

5.3.4.2 Explanation BLEU

The BLEU score measures how similar the prediction is to the reference at sentence level as shown in Equation 5.1. BP stands for the Brevity Penalty, which penalizes the model for generating shorter sentence than the reference one. Formula 5.2 shows the Brevity Penalty factor where c is the length of the candidate sentence and r is the length of the reference sentence. w stands

Table 5.3: *Predefined CWE Groups*

Group Name	CWEs
Input Validation and Representation	CWE-20, CWE-22, CWE-74, CWE-79, CWE-89, CWE-116, CWE-1284, CWE-1287
Permissions and Privilege Management	CWE-264, CWE-266, CWE-269, CWE-272, CWE-284
Memory Management and Safety	CWE-119, CWE-120, CWE-125, CWE-787, CWE-416, CWE-415
Cryptographic Issues	CWE-310, CWE-327, CWE-328, CWE-329, CWE-320, CWE-321
Resource Management Errors	CWE-399, CWE-400, CWE-401, CWE-404, CWE-772
Information Exposure	CWE-200, CWE-201, CWE-209, CWE-312, CWE-319
Concurrency and Race Conditions	CWE-362, CWE-367, CWE-366, CWE-664
Injection Attacks	CWE-74, CWE-77, CWE-78, CWE-89, CWE-94, CWE-79

for weights used to address different types n-gram precision. N-gram is an N-character slice of a string, where N is the number of characters in the slice [11]. For example, the 2-gram of the string 'hello' is 'he', 'el','ll', 'lo'. The 4-gram of the string 'hello' is 'hell', 'ello'. By default, BLEU uses uniform weights of 1/4 since it calculates upto 4-grams. BLEU only needs a brevity penalty since the geometric mean of modified precision (p), which caps the precision counting scheme at the maximum number of n-grams in the reference, has discarded the n-gram occurances within lengthy predictions. Therefore, BLEU, ranging from 0 to 1, indicates how well the prediction n-grams match the n-grams in the reference with sentence length as close as possible to the reference [63]. For this task, the BLEU score measures how well the model generates the CWE explanations compared to the reference explanations.

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{5.1}$$

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ \exp\left(1 - \frac{r}{c}\right) & \text{if } c \leq r \end{cases} \tag{5.2}$$

5.3.4.3 Highlight Jaccard

The Jaccard formula for two sets A and B is the size of the intersection divided by the size of the union as expressed in Equation 5.3. The Jaccard score ranges from 0 to 1, where 0 means no overlap and 1 means complete overlap. In the case of the highlight Jaccard score, the two sets are the predicted highlights and the reference highlights. The Jaccard score measures how well the model generates the CWE highlights compared to the reference highlights.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.3}$$

5.4 Results

This section discusses the results of the experimental evaluation. The model results are the median scores of the available runs on the test dataset, ranging

Table 5.4: *The performance of the Llama3-Gradient model in the zero-shot, one-shot, few-shot, and many-shot settings.*

Setting	Average F1	Group F1	Explanation BLEU	Highlight Jaccard
Zero-Shot	0.0	0.524	0.0	0.0
One-Shot	0.162	0.574	0.025	0.044
Few-Shot	0.289	0.577	0.022	0.166
Many-Shot	0.322	0.587	0.019	0.126

from 3 runs to 9 runs depending on the model time cost. More runs could provide more confidence in the results but the hardware is also a factor to consider. It is an empirical process, which requires running multiple experiments, to determine the optimal number of runs for each model. However, since the focus is to explore the feasibility, not benchmarking the model performance, this chapter shows the general trends of the results rather than the specific numbers. For brevity, if the precision, recall, and F1 scores are the same, only the F1 score is reported.

5.4.1 ICL Setting Compare

Table 5.4 compares the model performance in different ICL settings and shows that best performance on CWE classification comes from many-shot setting while one-shot ICL has the best performance in explanation. Many-shot ICL performs the best in the highlight selection.

5.4.2 Case Study on CVE-2021-22991

CVE-2021-22991 is the first test sample in our dataset split. In this subsection we show the predictions and references of the zero-shot, one-shot, few-shot and many-shot settings to demonstrate the performance of the model on different ICL settings.

Table 5.5 shows the prediction of the Llama-3 model on CVE-2021-22991

in the zero-shot setting. The model classifies the CVE as CWE-20 without following the required output format. In Table 5.6, the model predicts the same CWE-20 as in the zero-shot setting but this time it follows the required output format even though there is no highlight selected for this CVE. Table 5.7 shows the prediction of the model on CVE-2021-22991 in the few-shot setting. The model correctly predicts CWE-119 for this CVE however it also predicts CWE-20, CWE-16, CWE-22, and CWE-79 which are not in the reference. The output format is followed but there are broken structure at the very beginning of the labels. Instead of no highlight, the model made up a huge highlight that is not from the original description. Table 5.8 shows the prediction of the model on CVE-2021-22991. The model correctly predicts CWE-119 for this CVE however it also predicts CWE-20 which is not in the reference. The predicted format also lacks properly formatted explanation and highlight.

From the case study, we can see that the model performs the best in the many-shot setting in terms of labeling the CWEs for the CVE input. However, the model still struggles to generate the correct explanation and highlight for the CVE input. Off the shelf, the model fails to understand the task. As demonstrations increase, the model gathers more meaningful information to make better predictions.

5.4.3 Effect of Rule Book

Table 5.9 shows the rules used to guide the model in making the predictions. Table 5.10 illustrates the effect of rules on the performance of the model. Rule 0 means that the model runs in the many-shot setting without any rules provided by the Rule Book. Rule 1 introduces the first rule to the model. The model decreased drastically in performance with only the first rule addressing too many CWE labels in the prediction since the model may pick labels that are relevant but different from the reference label. The first two rules together make the model performing better than with just the first rule. The model shows a steady increase in performance as more rules are introduced. The

inference time follows a different pattern though as the time cost varies with the number of rules without any clear tendency.

After introducing five rules into the Rule Book, the model achieves the best performance in terms of classification F1 score, explanation BLEU, and highlight Jaccard. The model still struggles to stop generating after the highlight and generates examples after the highlight in some cases even Rule 5 explicitly asks it to do the opposite. Overall, as the Rule Book provides more rules to the model the model performs better in predicting the CWE labels from the CVE descriptions. The quality of rules introduced into the RB determines the performance of the model.

5.5 Limitations

The Rule Book has provided significant performance boost to the model however largely depends on the quality of the rules. It is complicated to determine the quality of the rules therefore more advanced techniques such rule mining could be applied to generate the rules other than manual examination and rule making. The Rule Book is static and does not change during inference. This could be a limitation since the model could benefit from a dynamic Rule Book that changes according to the model performance.

Due to the small size of the dataset set, the increase in performance is limited in finetuning. Even though the model tested (Llama3-Gradient) has shown promising results in the CWE classification task, the model still struggles to generate the correct explanation and highlight for the CVE input at times. Therefore, some post-processing steps are needed to get the desired output. For instance, the model tends to generate examples even though the highlight section has been generated. However, most of the time, the model behaves as expected.

CVE references could provide key informations to a CVE. However, the CVE references are not always available for all CVEs and the references can

be very difficult to access and parse. Sometimes the references are behind vendor-locked pages or require a paid subscription to access.

The use of BLEU score as a metric for evaluating explanation quality has its limitations. BLEU primarily measures n-gram overlap between the generated explanation and a reference explanation, which may not fully capture the semantic correctness or relevance of the explanation. A better metric that considers semantic similarity or human judgment could provide a more comprehensive evaluation of explanation quality.

Table 5.5: *CVE-2021-22991 Prediction and Reference of Zero-Shot In-Context Learning*

CVE-2021-22991	On BIG-IP versions 16.0.x before 16.0.1.1, 15.1.x before 15.1.2.1, 14.1.x before 14.1.4, 13.1.x before 13.1.3.6, and 12.1.x before 12.1.5.3, undisclosed requests to a virtual server may be incorrectly handled by the Traffic Management Microkernel (TMM) URI normalization, which may trigger a buffer overflow, resulting in a DoS attack. In certain situations, it may theoretically allow bypass of URL based access control or remote code execution (RCE). Note: Software versions which have reached End of Software Development (EoSD) are not evaluated.
Prediction	Improper Restriction of Operations within the Bounds of a Memory Buffer and CWE-20: Use of a Particular Value of a Named Constant, Variable, or Field without a Valid Check.
Reference	CWE-119 := This CVE primarily involves a buffer overflow (CWE-119) within the URI normalization component of the Traffic Management Microkernel (TMM). This means an attacker could send specially crafted requests that exceed the buffer's capacity, potentially overwriting adjacent memory and leading to a denial of service (DoS) attack. — buffer overflow

Table 5.6: *CVE-2021-22991 Prediction and Reference of One-Shot In-Context Learning*

CVE-2021-22991	On BIG-IP versions 16.0.x before 16.0.1.1, 15.1.x before 15.1.2.1, 14.1.x before 14.1.4, 13.1.x before 13.1.3.6, and 12.1.x before 12.1.5.3, undisclosed requests to a virtual server may be incorrectly handled by the Traffic Management Microkernel (TMM) URI normalization, which may trigger a buffer overflow, resulting in a DoS attack. In certain situations, it may theoretically allow bypass of URL based access control or remote code execution (RCE). Note: Software versions which have reached End of Software Development (EoSD) are not evaluated.
Prediction	CWE-20 := The vulnerability allows for a buffer overflow, which can lead to a denial of service (DoS) attack or potentially remote code execution (RCE).
Reference	CWE-119 := This CVE primarily involves a buffer overflow (CWE-119) within the URI normalization component of the Traffic Management Microkernel (TMM). This means an attacker could send specially crafted requests that exceed the buffer's capacity, potentially overwriting adjacent memory and leading to a denial of service (DoS) attack. — buffer overflow

Table 5.7: *CVE-2021-22991 Prediction and Reference of Few-Shot In-Context Learning*

CVE-2021-22991	On BIG-IP versions 16.0.x before 16.0.1.1, 15.1.x before 15.1.2.1, 14.1.x before 14.1.4, 13.1.x before 13.1.3.6, and 12.1.x before 12.1.5.3, undisclosed requests to a virtual server may be incorrectly handled by the Traffic Management Microkernel (TMM) URI normalization, which may trigger a buffer overflow, resulting in a DoS attack. In certain situations, it may theoretically allow bypass of URL based access control or remote code execution (RCE). Note: Software versions which have reached End of Software Development (EoSD) are not evaluated.
Prediction	CWE-119 &&& CWE-20 &&& CWE-16 —— CWE-22 —— CWE-79 := This CVE is a complex description, highlighting multiple vulnerability types. The initial issue is a buffer overflow within the Traffic Management Microkernel (TMM) URI normalization, which can trigger a denial of service attack. Furthermore, in some situations, this flaw could potentially allow bypass of URL-based access control, as well as remote code execution (RCE). The description also mentions that certain versions of BIG-IP have reached End of Software Development (EoSD), which means they are no longer being evaluated or patched for security issues. — undisclosed requests to a virtual server may be incorrectly handled by the Traffic Management Microkernel (TMM) URI normalization, which may trigger a buffer overflow, resulting in a DoS attack. In certain situations, it may theoretically allow bypass of URL based access control or remote code execution (RCE).
Reference	CWE-119 := This CVE primarily involves a buffer overflow (CWE-119) within the URI normalization component of the Traffic Management Microkernel (TMM). This means an attacker could send specially crafted requests that exceed the buffer's capacity, potentially overwriting adjacent memory and leading to a denial of service (DoS) attack. — buffer overflow

Table 5.8: *CVE-2021-22991 Prediction and Reference of Many-Shot In-Context Learning*

CVE-2021-22991	On BIG-IP versions 16.0.x before 16.0.1.1, 15.1.x before 15.1.2.1, 14.1.x before 14.1.4, 13.1.x before 13.1.3.6, and 12.1.x before 12.1.5.3, undisclosed requests to a virtual server may be incorrectly handled by the Traffic Management Microkernel (TMM) URI normalization, which may trigger a buffer overflow, resulting in a DoS attack. In certain situations, it may theoretically allow bypass of URL based access control or remote code execution (RCE). Note: Software versions which have reached End of Software Development (EoSD) are not evaluated.
Prediction	CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer and CWE-20: Use of a Particular Value of a Named Constant, Variable, or Field without a Valid Check.
Reference	CWE-119 := This CVE primarily involves a buffer overflow (CWE-119) within the URI normalization component of the Traffic Management Microkernel (TMM). This means an attacker could send specially crafted requests that exceed the buffer's capacity, potentially overwriting adjacent memory and leading to a denial of service (DoS) attack. — buffer overflow

Table 5.9: *Rules in the Rule Book*

ID	Content
1	Avoid overgeneralizing with multiple CWEs (e.g., CWE-78, CWE-1299, CWE-190, CWE-25) when a more specific CWE like CWE-119 is evident.
2	The prediction should avoid CWE entries unrelated to the exploit mechanism.
3	Give output precisely in the format CWE_label := Explanation — highlight. Do not give examples.
4	Make sure highlight the input description in the end of the prediction, starting with the '—' character after the explanation.
5	Stop generating after highlights; do not generate Examples.
6	Focus on assigning a specific CWE rather than assigning multiple CWEs. Prioritize Correct CWE Classification: If the vulnerability stems from incorrect type conversion, CWE-704 (Incorrect Type Conversion or Cast) should be used instead of CWE-190 (Integer Overflow) and CWE-476 (NULL Pointer Dereference) unless explicitly mentioned in the vulnerability details.
7	Prioritize Correct CWE Classification: If the vulnerability stems from incorrect type conversion, CWE-704 (Incorrect Type Conversion or Cast) should be used instead of CWE-190 (Integer Overflow) and CWE-476 (NULL Pointer Dereference) unless explicitly mentioned in the vulnerability details.
8	Limit CWE Selection to Relevant Entries: Avoid excessive CWE assignments. Predictions should focus only on CWEs directly relevant to the vulnerability description, typically one to three CWE entries unless multiple distinct exploit mechanisms are involved.
9	Generate unique CWEs in the prediction; do not repeat the same CWE in the output.

Table 5.10: *The performance of the Llama3-Gradient model with and without the Rule Book.*

Rules	Precision	Recall	F1	E. BLEU	H. Jaccard	Time(s)
0	0.322	0.322	0.322	0.019	0.126	**541.7**
1	0.213	0.213	0.213	**0.034**	0.084	1166.9
2	0.257	0.257	0.257	0.015	0.046	1438.5
3	0.264	0.264	0.264	0.022	0.140	1371.7
4	0.293	0.293	0.293	0.021	**0.158**	1059.6
5	0.380	0.380	0.380	0.031	0.144	1288.4
6	0.289	0.289	0.289	0.022	0.107	1424.4
7	0.385	0.385	0.385	0.026	0.086	983.9
8	**0.418**	**0.418**	**0.418**	0.030	0.121	1427.8
9	0.333	0.333	0.333	0.017	0.113	1550.1

Chapter 6

Conclusion

This thesis has explored the integration of AI-driven methods for vulnerability assessment, specifically targeting the challenges of identifying poorly reported vulnerabilities, estimating their severity, and improving classification accuracy. We have demonstrated that AI models, particularly Large Language Models (LLMs), can significantly enhance cybersecurity workflows by automating and improving the consistency of vulnerability analysis.

The proposed methodology (the USE model) for Common Vulnerability Scoring System (CVSS) estimation in Chapter 3 has proven to be more accurate and consistent than human expert assessments. The AI-driven models exhibit reduced estimation error and lower variance in CVSS score prediction, highlighting their potential as a viable alternative to manual evaluation. The USE model achieves 0.73 accuracy in predicting CVSS v3.1 base score and 0.128 mean squared error, which is 4 times lower than the human expert group (0.537). The model retains its performance even when evaluated on unseen vulnerabilities, demonstrating its generalizability and robustness.

Furthermore, one of the key contributions of this work is the development of an AI-based approach for assessing the quality of vulnerability descriptions as shown in Chapter 4. By leveraging machine learning techniques, this study has shown that AI models can identify poorly reported vulnerabilities with high precision, aiding in the standardization of vulnerability descriptions.

Additionally, this PhD study has made progress in the automated classification of vulnerabilities as shown in Chapter 5. By refining in-context learning techniques and incorporating a rule-based approach, this study has improved the classification accuracy of vulnerabilities while ensuring explainability and interpretability. These improvements address the longstanding issue of inconsistent and subjective vulnerability labeling, providing a scalable and reproducible framework for cybersecurity applications. The many-shot in-context learning setting with achieves 0.322 F1 score in classifying CWEs from CVE descriptions.

From a broader perspective, this work contributes to the intersection of AI and cybersecurity by demonstrating how AI-driven solutions can mitigate inconsistencies in vulnerability reporting, enhance risk assessment methodologies, and streamline cybersecurity threat analysis. The findings from this research provide a foundation for further advancements in AI-assisted cybersecurity, including the development of more robust, explainable, and scalable models.

This PhD research has provided valuable insights into the integration of AI-driven methods for vulnerability assessment, contributing to the advancement of human-centric cybersecurity practices. By deploying an automated AI-based vulnerability assessment framework, this work has the potential to significantly increase the processing efficiency of vulnerability management systems, reduce human error, and enhance the overall security posture of organizations. As cyberwarfare and digital threats continue to evolve, the need for robust, scalable, and explainable AI solutions in cybersecurity becomes vital to safeguarding critical infrastructure and sensitive data.

Future Work

This section discusses the future work that could be done based on the findings of this thesis.

Explainable CVSS A larger number of expert samples could help to reduce the variance in human estimation and therefore provide a more accurate comparison between the AI model and the human experts. The AI models also lacks the ability to explain the reasoning behind the CVSS score prediction. Provision of the reasoning behind the prediction would be necessary to build trust in the AI model. It is also worth exploring the performance of the AI model on the upcoming CVSS 4.0 scoring system and how effective the AI models are in helping to improve the vulnerability processing pipeline.

Data Improvements and Online Model A more extensively domain expert reviewed group effort could improve both the quality and the quantity of the labeled data. As LLMs have shown strong few-shot learning capabilities [10], an online EVAL model could be trained on the fly through a few of instructions and examples, and then be used to classify the incoming vulnerability reports in real time.

Rule Mining As shown in Chapter 5, the Rule Book has provided significant performance boost to the model however largely depends on the quality of the rules. Improving the quality of these rules could be a promising direction for future work. By carefully crafting the rules, the model can be guided to make better predictions.

Dynamic Rule Book The Rule Book as in Chapter 5 is static and does not change during inference. For future work, a dynamic Rule Book could be developed that injects information dynamically into the model.

CVE References A future work looking into the CVE references could be a good way to improve the performance of the model. References often include key information for the CVE. But the references are not always available for all CVEs and the references can be very difficult to access.

Unified Framework Moving forward, developing a unified framework that integrates real-time vulnerability detection, continuous learning from new cyber threat data, and dynamic model adaptation will be critical. This framework should not only improve predictive accuracy and interpretability but also address the evolving nature of cybersecurity risks by incorporating meta-learning strategies and scalable rule-based systems that work synergistically.

Model Finetuning With the increasing availability of computational resources and advanced techniques, finetuning the models with multiple runs and hyperparameter optimization could further improve the model performance on the tasks discussed in this thesis and give a more conclusive evaluation of the model capabilities.

Appendix A

Co-Authorship Form

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Co-Authorship Form

School of Graduate Research
The University of Waikato
Private Bag 3105
Hamilton 3240, New Zealand
Phone +64 7 838 5096
Email: SGR@waikato.ac.nz
Website: http://www.waikato.ac.nz/students/research-degree

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in your appendices for all the copies of your thesis submitted for examination and library deposit (including digital deposit).

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 3 are extracted from Journal Paper Ai-enabled automated common vulnerability scoring from common vulnerabilities and exposures descriptions in International Journal of Information Security, Volumne 24, Issue 1, Page 1-20, Springer Berlin Heidelberg Published date: 2025/2	
Nature of contribution by PhD candidate	Conducted the experiments and writing up the paper
Extent of contribution by PhD candidate (%)	85

CO-AUTHORS

Name	Nature of Contribution
Dr. Vimal Kumar	Supervision and Peer Review
Dr. Bernhard Pfahringer	Supervision and Peer Review
Dr. Albert Bifet	Supervision and Peer Review

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

Name	Signature	Date
Vimal Kumar		30/4/2025
Bernhard Pfahringer		30/4/2025
Albert Bifet		01/5/2025

Appendix B

Acknowledgements

The Ministry of Business, Innovation, and Employment (MBIE) supported my PhD study under the Catalyst: Strategic Cyber Security Research Programme Grant. I conducted all research presented in this thesis independently, ensuring the integrity and originality of the findings. Dr.Kumar has supervised all three papers during my PhD study. Dr.Mayo has provided the guidance for the first paper. Dr.Pfahringer has supervised the second paper and the third paper and as my main supervisor, he guided me through not only the research but also the mental challenges that I faced during my PhD study. Dr. Bifet has provided the infrastructure needed throughout the three papers. I have used generative AIs to help me overcome language barriers and generating LaTeX typeseting syntex in this thesis.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. FLAIR: An easy-to-use framework for state-of-the-art NLP. In Waleed Ammar, Annie Louis, and Nasrin Mostafazadeh, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[4] Dzmitry Bahdanau. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] Paige Bailey, Paul Mooney, Ashley Chow, and Addison Howard. Google - gemini long context. `https://kaggle.com/competitions/gemini-long-context`, 2024. Kaggle.

[6] Markus Bayer, Philipp Kuehn, Ramin Shanehsaz, and Christian Reuter. Cysecbert: A domain-adapted language model for the cybersecurity domain. *ACM Transactions on Privacy and Security*, 27(2):1–20, 2024.

[7] Alexander Beck and Stefan Rass. Using neural networks to aid cvss risk aggregation—an empirically validated approach. *Journal of Innovation in Digital Ecosystems*, 3(2):148–154, 2016.

[8] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[9] Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 interactive presentation sessions*, pages 69–72, 2006.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, volume 161175, page 14. Ann Arbor, Michigan, 1994.

[12] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder for english. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pages 169–174, 2018.

[13] Guangyi Chen, Tianren Zhang, Jiwen Lu, and Jie Zhou. Deep meta metric learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9547–9556, 2019.

[14] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[15] CVE Numbering Authorities CNA. Key details phrasing - cve, 2023.

[16] Emily Coffey. Cybersecurity company plans to add over 100 six-figure jobs in colorado springs, Feb 2024.

[17] Joana Cabral Costa, Tiago Roxo, João BF Sequeiros, Hugo Proença, and Pedro RM Inácio. Predicting cvss metric via description interpretation. *IEEE Access*, 2022.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[19] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1994.

[20] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, 2024.

[21] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[22] Explosion. Span categorization prodigy an annotation tool for ai, machine learning and nlp.

[23] Li Fe-Fei et al. A bayesian approach to unsupervised one-shot learning of object categories. In *proceedings ninth IEEE international conference on computer vision*, pages 1134–1141. IEEE, 2003.

[24] FIRST. Common vulnerability scoring system version 3.1: Specification document, Jun 2023.

[25] Tim Genewein, Grégoire Delétang, Anian Ruoss, Li Kevin Wenliang, Elliot Catt, Vincent Dutordoir, Jordi Grau-Moya, Laurent Orseau, Marcus Hutter, and Joel Veness. Memory-based meta-learning on non-stationary distributions. In *International conference on machine learning*, pages 11173–11195. PMLR, 2023.

[26] Hassan Gharoun, Fereshteh Momenifar, Fang Chen, and Amir H Gandomi. Meta-learning approaches for few-shot learning: A survey of recent advances. *ACM Computing Surveys*, 56(12):1–41, 2024.

[27] Adrian Gonzalez, Alec Summers, Alicia Gillum, Charles Schmidt, Chris Coffin, Connor Mullaly, David Rothenberg, Gage Hackford, Gananand Kini, John DeCarlo, Jordan Burton, Kent Sanders, Luke Malinowski, O'Ryan Lattin, Rich Piazza, Robert L. Heinemann, Rushi Purohit, Steve

Coley, Christopher Turner, and David Jung. Cwe - 2023 cwe top 25 most dangerous software weaknesses, Jul 2023.

[28] google gemini. Github - google-gemini/generative-ai-python: The official python library for the google gemini api, Oct 2024.

[29] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`, 2022.

[30] Hao Guo, Sen Chen, Zhenchang Xing, Xiaohong Li, Yude Bai, and Jiamou Sun. Detecting and augmenting missing key aspects in vulnerability descriptions. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(3):1–27, 2022.

[31] Jacob Haislip, Kalin Kolev, Robert Pinsker, and Thomas Steffen. The economic cost of cybersecurity breaches: A broad-based analysis. In *Workshop on the economics of information security (WEIS)*, volume 9, 2019.

[32] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[33] Hannes Holm and Khalid Khan Afridi. An expert-based investigation of the common vulnerability scoring system. *Computers & Security*, 53:18–30, 2015.

[34] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[35] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 1681–1691, 2015.

[36] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[37] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[38] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[39] Nishtha Kesswani and Sanjay Kumar. Maintaining cyber security: Implications, cost and returns. In *Proceedings of the 2015 ACM SIGMIS Conference on Computers and People Research*, pages 161–164, 2015.

[40] Ashraf M Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. Multinomial naive bayes for text categorization revisited. In *Australasian Joint Conference on Artificial Intelligence*, pages 488–499. Springer, 2004.

[41] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[42] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, pages 1–30. Lille, 2015.

[43] Oliver Kramer. Scikit-learn. In *Machine learning for evolution strategies*, pages 45–53. Springer, 2016.

[44] Bartlomiej Jacek Kubica, Olga Kosheleva, and Vladik Kreinovich. Mcfadden's discrete choice and softmax under interval (and other) uncertainty: Revisited. 2024.

[45] Philipp Kuehn, Markus Bayer, Marc Wendelborn, and Christian Reuter. Ovana: An approach to analyze and improve the information quality of vulnerability databases. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–11, 2021.

[46] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[47] Jinhyuk Lee, Anthony Chen, Zhuyun Dai, Dheeru Dua, Devendra Singh Sachan, Michael Boratko, Yi Luan, Sébastien MR Arnold, Vincent Perot, Siddharth Dalmia, et al. Can long-context language models subsume retrieval, rag, sql, and more? *arXiv preprint arXiv:2406.13121*, 2024.

[48] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7871–7880, 2020.

[49] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[50] Bob Martin, Mason Brown, Alan Paller, Dennis Kirby, and S Christey. Cwe. *SANS top*, 25, 2011.

[51] Daniel McFadden. Conditional logit analysis of qualitative choice behavior. 1972.

[52] Peter Mell, Karen Scarfone, Sasha Romanosky, et al. A complete guide to the common vulnerability scoring system version 2.0. In *Published by FIRST-forum of incident response and security teams*, volume 1, page 23, 2007.

[53] Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[54] Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[55] Taro Miyazaki, Hideya Mino, and Hiroyuki Kaneko. Understanding how positional encodings work in transformer model. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 17011–17018, 2024.

[56] Stephan Neuhaus and Thomas Zimmermann. Security trend analysis with cve topic models. In *2010 IEEE 21st International Symposium on Software Reliability Engineering*, pages 111–120. IEEE, 2010.

[57] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[58] Maciej Nowak, Michał Walkowski, and Sławomir Sujecki. Machine learning algorithms for conversion of cvss base score from 2.0 to 3. x. In *International Conference on Computational Science*, pages 255–269. Springer, 2021.

[59] NIST NVD. National vulnerability database, 2022.

[60] John W Oller Jr. Cloze tests of second language proficiency and what they measure 1. *Language learning*, 23(1):105–118, 1973.

[61] Ronan Oostveen. Cwe-assist: A framework for automating cwe classification. Master's thesis, University of Twente, 2024.

[62] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. *Advances in neural information processing systems*, 22, 2009.

[63] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

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

[65] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[66] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[67] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[68] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[69] James Rundle. Boards still lack cybersecurity expertise. *WSJ*, Sep 2023.

[70] Jukka Ruohonen. A look at the time delays in cvss vulnerability scoring. *Applied Computing and Informatics*, 15(2):129–135, 2019.

[71] Mustafizur R Shahid and Hervé Debar. Cvss-bert: Explainable natural language processing to determine the severity of a computer security vulnerability from its description. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1600–1607. IEEE, 2021.

[72] Morgan Smith. Companies have an incredible need for this in-demand skill, says google exec—and it pays over 100,000 a year, Nov 2023.

[73] Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. Fast wordpiece tokenization. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 2089–2103, 2021.

[74] Yisheng Song, Ting Wang, Puyu Cai, Subrota K Mondal, and Jyoti Prakash Sahoo. A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities. *ACM Computing Surveys*, 55(13s):1–40, 2023.

[75] Ferda Özdemir Sönmez. Classifying common vulnerabilities and exposures database using text mining and graph theoretical analysis. In *Machine Intelligence and Big Data Analytics for Cybersecurity Applications*, pages 313–338. Springer, 2020.

[76] Jonathan Spring, Eric Hatleback, Allen Householder, Art Manion, and Deana Shick. Time to change the cvss? *IEEE Security & Privacy*, 19(2):74–78, 2021.

[77] Jonathan Spring, Eric Hatleback, A Manion, and D Shic. Towards improving cvss. *Software Engineering Institute, Carnegie Mellon University, Tech. Rep*, 2018.

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

[79] Kensuke Sumoto, Kenta Kanakogi, Hironori Washizaki, Naohiko Tsuda, Nobukazu Yoshioka, Yoshiaki Fukazawa, and Hideyuki Kanuka. Automatic labeling of the elements of a vulnerability report cve with nlp. In

2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI), pages 164–165. IEEE, 2022.

[80] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[81] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[82] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[83] Inacio Vieira, Will Allred, Séamus Lankford, Sheila Castilho, and Andy Way. How much data is enough data? fine-tuning large language models for in-house translation: Performance evaluation across multiple dataset sizes. In Rebecca Knowles, Akiko Eriguchi, and Shivali Goel, editors, *Proceedings of the 16th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 236–249, Chicago, USA, September 2024. Association for Machine Translation in the Americas.

[84] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[85] Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136, 2023.

[86] Yasuhiro Yamamoto, Daisuke Miyamoto, and Masaya Nakayama. Text-mining approach for estimating vulnerability score. In *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, pages 67–73. IEEE, 2015.

[87] Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When scaling meets llm finetuning: The effect of data, model and finetuning method. *arXiv preprint arXiv:2402.17193*, 2024.

[88] Siqi Zhang, Mengyuan Zhang, and Lianying Zhao. Viet: A tool for extracting essential information from vulnerability descriptions for cvss evaluation. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 386–403. Springer, 2023.

[89] Zijing Zhang, Vimal Kumar, Michael Mayo, and Albert Bifet. Assessing vulnerability from its description. In *International Conference on Ubiquitous Security*, pages 129–143. Springer, 2023.

[90] Zijing Zhang, Vimal Kumar, Bernhard Pfahringer, and Albert Bifet. Ai-enabled automated common vulnerability scoring from common vulnerabilities and exposures descriptions. *International Journal of Information Security*, 24(1):1–20, 2025.