What Matters Most in Vulnerabilities? Key Term Extraction for CVE-to-CWE Mapping with LLMs

Stefano Simonetto[0009−0009−9778−4019],
Ronan Oosteven, Thijs Van Ede, Peter Bosch, and Willem Jonker

University of Twente, Enschede, The Netherlands
{s.simonetto, r.oostveen, t.s.vanede, p.bosch, w.jonker}@utwente.nl

Abstract. Mapping Common Vulnerabilities and Exposures (CVEs) to the related Common Weakness Enumerations (CWE) is crucial in cybersecurity because this link allows categorizing vulnerabilities, prioritizing remediation efforts, and improving mitigation strategies. However, this process often requires manual effort to understand and connect a CVE to its CWE, which is undesirable due to its time-consuming nature. Existing automated approaches typically rely on vectorizing CVE descriptions using embedding techniques followed by machine learning classifiers. However, little attention has been given to evaluating whether CVE descriptions are the most effective starting point for automated classification, as these descriptions often contain irrelevant details that do not contribute to CVE-to-CWE mapping. Our research investigates to what extent we can automatically extract key information from CVE descriptions required to perform such mapping and evaluates how this technique improves existing methods for identifying related weaknesses using this extracted information. To this end, we present an approach that automates extracting key terms of CVEs through Large Language Models and evaluate the effect of focusing on various parts of the CVE description. We show that our key term extraction technique improves the F1-score of transformer-based classification of CVEs into CWEs up to 8.88% while providing a modest increase for more traditional mapping approaches.

Keywords: CVE · CWE · mapping · key-term extraction

1 Introduction

In cybersecurity, mapping Common Vulnerabilities and Exposures (CVEs)[1] to their corresponding Common Weakness Enumerations (CWEs)[2] is a crucial yet challenging task. Knowing the CWE associated with a specific CVE is central to the threat analysis. The intricate nature of CVE descriptions, which often detail specific products and versions, makes mapping CVEs to CWEs a complex task. This process requires a higher level of abstraction, as CWEs focus on the root cause and type of vulnerability, which is not directly tied to specific components or vendors that are present in the CVE description.

[1] https://cve.mitre.org/
[2] https://cwe.mitre.org/

From the beginning of 2025, an average of 130 new CVEs are added to the database each day [1], with every entry requiring manual processing by CVE Numbering Authorities (CNAs). This labor-intensive process is compounded by the complexity of the CWE taxonomy, which demands careful selection from a broad range of categories. As shown in Figure 6 in the Appendix, experts struggle to assign appropriate CWE labels to new CVEs promptly. Accurately labeling CWEs is essential as it enables the clustering of related CVEs and provides valuable insights into the patterns exploited by threat actors.

The existing literature on CVE-to-CWE mapping typically begins with the CVE description, which is then transformed into a vector representation using various techniques such as TF-IDF [2], Text-CNN [24], Bag-of-Words [3], and transformer-based [18]. These vectors are subsequently classified using different machine-learning models. Our research question concerns a preprocessing step that extracts the key terms from a description to allow the creation of better embeddings that improve subsequent classification. We answer the following: "Which key terms from a CVE description are most useful for accurately mapping it to the corresponding CWE?". By understanding what type of information is essential for the mapping, we can improve the existing model's prediction performance and better understand how to write clear CVE descriptions that communicate what weaknesses are exposed by the vulnerability.

To tackle this challenge, the paper describes a novel methodology to automate the extraction of key terms for CVE representation and compare CVE sections to determine which parts classifiers should focus on. Existing tools focus on extracting the subject or general keywords of a text. However, for CVEs, this information often involves specific products and version numbers, which are insignificant when determining the corresponding CWE. Instead, we prioritize extracting only the key terms of the CVE useful for this mapping. We investigate the use of LLMs (Llama and GPT) for automatically extracting key terms (focusing on fundamental aspects, context, and impact) of the vulnerability description.

The contributions of this paper are as follows. First, we propose a structured approach for extracting key terms from CVE descriptions to enhance CVE-to-CWE mapping. Leveraging large language models and in-context learning, our method distills the essential semantic elements (namely, the core aspects, context, and impact) while omitting product-specific details that are generally irrelevant for CWE classification.

Second, we perform a detailed manual evaluation of the extracted key terms and conduct a thorough comparative analysis of various CVE representations. Our findings demonstrate that our method consistently improves mapping performance, achieving up to an 8.88% increment in the F1-score on transformer-based models. Notably, these performance gains hold consistently across both unsupervised and supervised learning frameworks, as well as when applied to more conventional mapping techniques.

To foster collaboration in this area, we have made our code and results publicly available[3].

[3] https://anonymous.4open.science/r/CVEtermsextraction/README.md

2 Background

CVEs enable security professionals and organizations to communicate effectively about specific weaknesses, ensuring universal reference to vulnerabilities by a common name. CVEs are indispensable for facilitating knowledge-sharing and fostering collaboration among researchers and vendors to develop appropriate patches or mitigations, thus safeguarding systems from potential exploits.

Common Weakness Enumeration is a community-developed catalog of common software weaknesses and security flaws. Unlike CVEs, which pinpoint specific vulnerabilities, CWEs categorize broader classes of weaknesses, encompassing various instances of similar vulnerabilities.

Over the years, annual CVE reports have surged dramatically, from 1,579 in 1999 to over 38,000 in 2024. At the same time, a persistent and widening gap in mapping CVEs to CWEs highlights the urgent need for a rapid, reliable, and automated classification system that minimizes reliance on human intervention. This trend is illustrated in Figure 6 in the Appendix.

2.1 Motivation

In real-world scenarios, threat actors sequentially exploit multiple vulnerabilities to breach systems [13]. To identify these attack sequences, extracting the root cause at each step is crucial, uncovering the specific weaknesses that enable progression through the exploit. These causes are systematically identified through the CWE framework, which provides a clear understanding of the underlying weaknesses. Accurately recognizing these root causes is crucial, as it directly informs where investments, policies, and practices should be focused to effectively eliminate vulnerabilities at their source [18].

Ideally, CVE descriptions use a specified template that allows us to extract the root cause easily and adhere to a structured format, such as the one proposed in [6] and highlighted in Fig. 1:

-[VULNTYPE] in [COMPONENT] of [VENDOR] [PRODUCT] [VERSION] allows [ATTACKER] to [IMPACT] via [VECTOR].
-[COMPONENT] in [VENDOR] [PRODUCT] [VERSION] has a [ROOT CAUSE], which enables [ATTACKER] to [IMPACT] via [VECTOR].

Fig. 1: Vulnerability Representation Templates

Unfortunately, as noted by [21], the use of templates in vulnerability descriptions has declined significantly, from 99.6% in 2010 to 58.5% in 2021. This suggests the need for a more flexible yet informative format that maintains the essential elements while adapting to evolving semantic changes in CVE descriptions.

To address this issue in vulnerability descriptions, we focus on NLP methods to be flexible in mapping even where no template format is available. Our preprocessing method automatically extracts key terms from CVEs, capturing the core aspects, contextual details, and potential impact of each vulnerability.

2.2 CVE to CWE linking example

The CVE description is precise toward a specific product or version, providing detailed information on how it affects the system. As an example, consider the description of CVE-2013-2762 as follows:

> "The Schneider Electric Magelis XBT HMI controller has a default password for authentication of configuration uploads, which makes it easier for remote attackers to . . . "

The CVE description focuses on the specific product (subject), and contrasts with the broader context required for fitting it into CWE classifications. However, when mapping to a CWE classification, the focus shifts toward the broader underlying weakness rather than the particular product details. While product names and versions are essential for describing a specific vulnerability, they offer little value in understanding the broader weakness category represented by a CWE.

2.3 Hierarchical CWE structure

CWEs are structured in a parent-child relationship, where generic weaknesses (e.g., improper restriction of operation within the bounds of a Memory Buffer) have more specific child weaknesses (e.g., Out-of-bounds Read vs Out-of-bounds Write). This structure represents the generalization-specialization relationship as pointed out in [12]: the more elevated the position in the hierarchy, the greater the abstraction; in contrast, the lower the position, the more focused on specifics. To illustrate this concept and highlight the related challenge in the classification, consider the following hierarchy within the CWE framework:

- **Class:** Improper Restriction of Operations within the Bounds of a Memory Buffer
  - **Base:** Out-of-bounds Write
  - **Base:** Out-of-bounds Read
    - **Variant:** Buffer Under-read
    - **Variant:** Buffer Over-read

The challenge lies in extracting the differentiators between each classification option. Our method effectively identifies the nuances that differentiate one CWE from its parent or siblings, such as distinguishing between buffer overflow-read and buffer overflow-write. The CWE framework follows this hierarchical structure, as illustrated in the CWE-1003 view. The specification emphasizes that each CVE should be assigned the most specific, lowest-level label within the hierarchy whenever possible.

3 Related work

Recent studies have explored two avenues to improve CVE-to-CWE mapping. One approach is to adopt better CVE description embedders, ranging from TF-IDF [4] to text-to-vec [16] and transformer-based methods [23]. The other approach focuses on mapping these embeddings to the correct CWE label through multi-stage processes.

In our work, we focus on the former, specifically on enhancing the embedding process by targeting only the keywords extracted from a CVE description rather than

embedding the entire text. As demonstrated by [20], this keyword-focused embedding can significantly improve quality, since averaging embeddings over all words can dilute the meaning of the critical terms. To identify these significant keywords, key phrase extraction methods such as YAKE [5] and KeyBERT [11] have been applied. YAKE is an unsupervised and lightweight technique that determines term relevance based on statistical features within a document, whereas KeyBERT leverages BERT embeddings to extract keywords that best match the document's context, thereby providing a more context-aware extraction process.

Although these keyword extractors highlight the most important terms in the CVE text, they do not consider the specific application of those terms, such as CVE-to-CWE mapping. In this scenario, terms such as the CVE subject might be extracted as keywords by YAKE or KeyBERT but are not particularly helpful for accurately mapping CVEs to CWEs. Therefore, task-specific considerations are crucial to improving the usefulness of extracted keywords in specialized contexts such as CVE classification.

4 Dataset

Key term extraction can be applied to any CVE description, but evaluating its benefits requires a meticulously prepared dataset for both machine learning training and testing. To assess the extent to which key term extraction enhances the automated mapping of CVE descriptions to CWEs, we use NIST's National Vulnerability Database (NVD). The NVD provides a manually verified mapping of all vulnerabilities to the weaknesses listed in the default "CWE-1003 view."

The dataset was prepared through the following steps. First, we converted multi-label CVEs, comprising 1.5% of the dataset, into single-label instances by duplicating entries, increasing the total from 222,240 to 225,886. Next, we applied a minimum sample threshold to focus on prevalent CWEs to cover 95% of all CVEs, narrowing our focus to 57 classes. Finally, we split the data so that 10% of the original dataset forms the test set (12,845 samples) and the remaining data (comprising of the 57 classes) is divided into 90% for training and 10% for validation. For the training and validation sets, to mitigate bias from classes with excessively large sample sizes (e.g., CWE-79 with over 25,000 samples), we undersampled classes with more than 3,000 samples, resulting in a balanced set of 72,372 CVE descriptions with no overlap between the test and training-validation sets.

5 Preliminary analysis

We conducted a comparison of two state-of-the-art key term extraction methods, YAKE and KeyBERT, to assess their effectiveness in extracting key terms that are useful for mapping CVEs to CWEs. The objective is to determine how well these tools could identify key terms relevant to this specific type of mapping while avoiding extracting terms that are simply the subject of the CVE, which are often not useful for this purpose.

To evaluate these tools, we selected two CVE descriptions for each CWE (114 examples in total) and extracted the top three key terms using both YAKE and KeyBERT. The results were manually analyzed to determine whether the retrieved terms provided useful information ('root cause' and 'vulnerability type') for CVE-to-CWE mapping and to what extent they included terms that merely represented the subject of the CVE.

YAKE identified the subject as a key term 74.56% of cases and provided useful information 47.37% of the time. On the other hand, KeyBERT detected the subject even more frequently (92.11%) and produced significantly fewer useful CWE-relevant terms (14.04%).

Overall, our comparison shows that YAKE offers a better balance in terms of avoiding overly specific subject-related terms and providing terms more directly related to the CWE. Both YAKE and KeyBERT exhibit significant limitations, which motivated us to experiment with our approach based on in-context learning leveraging LLMs to improve the accuracy and relevance of key term extraction for CVE-to-CWE mapping.

6 Term extraction methodology

This section details our approach for extracting key terms from CVE descriptions leveraging LLMs as illustrated in Fig. 2. In Subsection 6.2, we explore the role of LLMs in key term extraction and conduct a case study to evaluate the impact of different models on our approach.

6.1 Term extraction

In this section, we develop a pipeline for automatically extracting key terms from CVE descriptions, streamlining the process by avoiding the involvement of expert revision. The focus is on abstracting the CVE description from unrelated details to establish a direct mapping from CVE to CWE.

We define the following categories based on the standardized template that every vulnerability should adhere to, as shown in Fig. 1:

- **Core terms** represent fundamental concepts that define the primary aspects or components of a vulnerability. These terms are essential for understanding the central issue and correspond to the 'Vulnerability Type' and 'Root Cause'.
- **Consequences** refer to the outcomes or results of actions. Consequences often describe the 'Impact' of a vulnerability being exploited.
- **Contextual terms** provide additional details or conditions that help clarify the circumstances under which a vulnerability occurs. These terms correspond to the 'Vector' or 'Attacker' attributes.

As shown in Fig. 2, we leverage in-context learning (ICL) [10] to extract key terms. ICL operates on the principle of learning from analogy by using a few demonstration examples to build a prompt context. This context is then combined with a query to form the input for the language model, which generates the prediction. As shown in [9], the transformer block implicitly modifies the weights of the MLP layer according to the context.

Given the challenging nature of the CVE-to-CWE mapping, finding the right prompt design is crucial for capturing the essence of the CVE, as described in the following subsections.

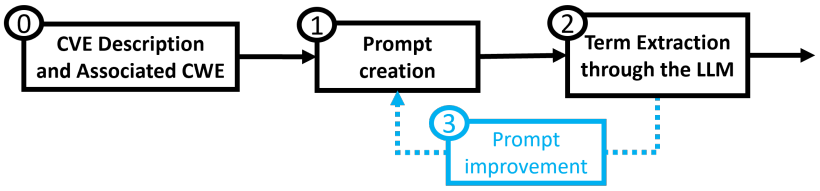

Fig. 2: Pipeline to extract key terms. **Note**: the blue dashed line represents a step used only during the prompt design phase. This step is omitted during inference, making the pipeline fully automated and constituted only by steps from 0 to 2.

1. Prompt creation Effective prompt design is essential for in-context learning, ensuring that the model fully understands the task requirements and produces meaningful outputs. We instructed the LLMs to extract three key elements from each CVE description (core terms, contextual terms, and consequences) while omitting extraneous details. Our final prompt includes an example illustrating these elements, derived from a typical CVE description that contains both relevant and irrelevant information. This example helps guide the LLM in prioritizing the most important aspects. The full prompt is provided in Appendix 12.2.

2. Term extraction through the LLM Our process summarizes CVE descriptions to their core components and outputs them in JSON format. For this task, we experimented with GPT-3.5 Turbo and LLaMA 2 (7B), which are referred to throughout the paper as **GPT** and **LLaMA**, respectively.

Notably, while GPT produced outputs that were immediately parsable as JSON, only 19% of Llama's outputs met this criterion. The remaining outputs required additional post-processing to achieve a valid JSON format, though regeneration was not necessary.

3. Prompt improvement In the design phase, we manually assessed the relevance of the extracted terms to ensure they accurately reflect the provided CVE descriptions. This manual review is crucial for validating the effectiveness of the automated term extraction, as discussed in subsection 6.2. After multiple iterations, we refined the prompt, achieving the association of each CVE description with the three sets of representative terms.

As an example, initially, we included 'technical details' as a fourth item in the key terms list to maintain the specificity of the CVE description towards items such as 'Vendor', as shown in Figure 1. However, this approach led to the extraction of overly

specific terms focused solely on vulnerability, such as the product name. Consequently, we decided not to include these technical details, resulting in the final core terms, contextual terms, and consequences. Once the prompt is finalized, this step is no longer performed, leaving the key term extraction pipeline fully automated from steps 0 to 2.

6.2 Extracted terms manual evaluation

We manually validated 2 randomly selected CVE descriptions for each of the 57 CWE classes used in our evaluation, totaling 114 manually inspected samples.

Since there is no standardized benchmarking for evaluating LLM responses, as the assessment criteria vary based on the specific task and format, we evaluated the extracted terms from the CVE description based on three key attributes: hallucination, completeness, and consistency.

- Hallucination: This criterion checks whether the terms include any fabricated information not present in the original CVE description. Importantly, elaborations or clarifications of terms or concepts are not considered hallucinations. For example, in Table 5, if the short description mentions 'XSS' and the corresponding extracted core term is 'Cross-site scripting,' this is *not* treated as a hallucination. This metric is analogous to *1-precision*, ensuring that the extracted terms are grounded in the source text.
- Completeness: We assess whether the extracted terms capture all the essential aspects of the CVE description, ensuring no critical information is omitted. Rather than emphasizing the categorization of terms (e.g., core terms, contextual terms, consequences), we focus on providing a global representation of the CVE description. This ensures that the extracted terms reflect the full scope of the vulnerability. This metric is analogous to recall.
- Consistency: To measure consistency, we extracted key terms three times with the temperature set to the default value and compared the results to see if the relevant terms were maintained in each category. If the key terms in each category were preserved, the extraction was considered consistent; otherwise, it was classified as inconsistent.

An example for each criterion is proposed in Appendix 12.3.

Table 1: Llama and GPT comparison on hallucinations, completeness, and consistency. Consistency is measured across three extraction attempts, while hallucination and completeness are evaluated based on a single extraction.

Model	**Subject present in core terms**	**Hallucination**	**Completeness**	**Consistency**		
				Core	Contextual	Consequences
Llama	12.28%	15.78%	75.44%	82.46%	41.23%	64.06%
GPT	11.40%	6.14%	90.35%	97.37%	77.19%	87.72%

Table 1 clearly illustrates that the key terms extracted by GPT are more consistent, complete, and exhibit fewer hallucinations compared to those extracted by

Llama. Consequently, we selected GPT as our primary key term extractor in subsequent experiments (see Section 7). While certain aspects of CVE descriptions can be challenging to capture for both language models (as evidenced by the trends in both models), GPT consistently outperforms Llama in all key metrics.

The results of in-context learning using LLMs show substantial improvement compared to YAKE and KeyBERT. Specifically, GPT and Llama show a reduction in irrelevant subjects within the extracted terms, as well as significant advancements in extracting useful information for effective mapping. GPT and Llama both achieve far greater completeness compared to YAKE and KeyBERT, reinforcing the effectiveness of LLMs in this domain.

Table 1 shows that GPT produces significantly fewer hallucinations. This reduced hallucination rate directly impacts the precision of the extracted terms, ensuring they remain faithful to the original CVE descriptions. Additionally, GPT's higher completeness score ensures a more comprehensive representation of CVE information, minimizing the risk of leaving out critical details. Consistency across multiple runs is crucial for the reproducibility of experiments and future work. Even if the correct terms were consistently identified, they were labeled as inconsistent if placed in incorrect categories (e.g., core terms classified as contextual terms). Despite this categorization challenge, GPT demonstrated greater stability, providing more consistent term classification compared to Llama.

7 Terms comparison methodology

We analyze different focus areas of the CVE text, such as using the complete description, a combination of the extracted terms, and descriptions without the subject. This analysis determines which modification most effectively captures the semantic essence needed for accurate CVE to CWE mapping, as shown in Fig. 3. As discussed in Section 3, two main approaches exist to improve CVE-to-CWE mapping: developing better CVE description embedders and mapping these embeddings to the correct CWE label via multi-stage processes (e.g., multi-stage neural networks). Our work focuses on enhancing the first approach, and for the second step, we implemented a basic neural network, as achieving the best possible classification is not the primary goal of this study. To understand which representation of a CVE should be embedded, we run different experiments, considering how a CVE can be summarized.

1.Different scenarios

To analyze and optimize how CVE descriptions are processed, we propose a systematic approach to test various configurations of the text. These configurations are designed to address common issues with CVE descriptions and evaluate the impact of different term selections. The rationale behind each step is as follows, and an example for each scenario is provided in Table 2:

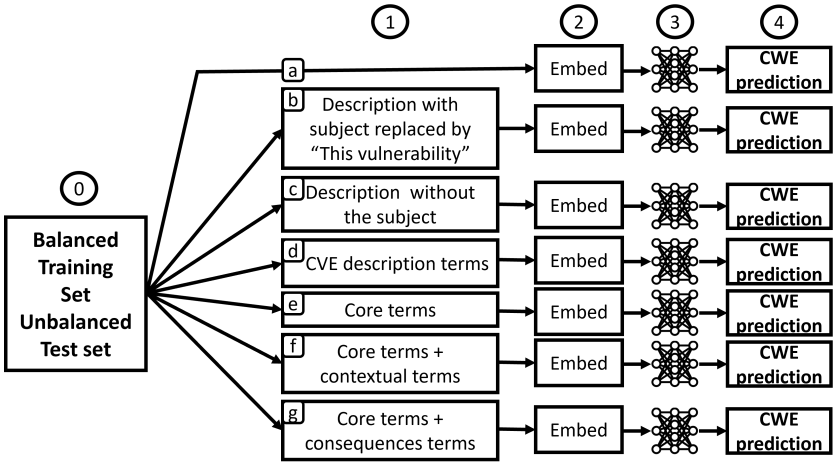

Fig. 3: Comparison of different CVE representations to select which is the best performing in the task of CVE to CWE mapping.

(a) Baseline Method

The simplest and widely adopted approach utilizes the entire CVE description. This method ensures that no information is omitted; however, CVE descriptions often include product-specific details that might limit generalization.

Replacing Subjects for Clarity

Many CVE descriptions start with product details, version numbers, and vendor information, making them less generalizable. To address this:

(b) Replace the subject (the portion of the sentence before the verb) with a generic term like "This vulnerability." This substitution aims to maintain sentence structure while removing product-specific identifiers.

(c) Replace the subject with a blank space to make the CVE description independent of any specific subject.

Term Extraction and Combination

Leveraging key terms extracted through a previously described pipeline, we conduct an ablation study to assess the best configuration:

(d) *Key terms*: Concatenate all extracted terms, including core, contextual, and consequence terms, to provide a comprehensive view of the CVE.

(e) Use only core terms.

(f) Combine core and contextual terms.

(g) Combine core and consequence terms.

Table 2: Description of CVE-2010-1459. In three scenarios, the subject (red) is a) left intact; b) replaced with "This vulnerability"; c) removed. Additionally, points d) to g) explore various combinations of core terms, contextual, and consequences: d) concatenate all of them, e) take only the core terms, f) combine core and contextual, and g) combine core and consequences.

CVE-2010-1459: The default configuration of ASP.NET in Mono before 2.6.4 has a value of FALSE for the EnableViewStateMac property, allowing remote attackers to conduct Cross-site Scripting (XSS) attacks via the __VIEWSTATE parameter to 2.0/menu/menu1.aspx. in the XSP sample project.

Core Terms	**Contextual Terms**	**Consequences**
Cross-site Scripting Vulnerability	Login form	Arbitrary HTML injection
Remote Attack	Remote attackers	Form parameter manipulation
Arbitrary Code Injection	Web script injection	Security compromise

NIST classification: CWE-79 (Cross-site Scripting)

2.Embed

As noted in Section 3, various embedding approaches have been proposed for mapping CVE text to CWE, with transformer-based embeddings emerging as the most promising. In our work, both the training and testing data are embedded using OpenAI's ada-embeddings, producing a 1536-dimensional dense vector that captures the semantic features of the text. Details on different transformer-based embedders can be found in Section 8.2.

3.Training

For the mapping step, we implemented a feed-forward neural network. This choice reflects our primary objective: to enhance the input representation for the downstream classifier (specifically, the embedding of CVE representation) rather than to replicate or benchmark state-of-the-art classification architectures. Notably, this design underpins the most recent and best-performing models in the literature [2, 7, 19], where only minor variations are introduced. Therefore, exploring these improvements was not within the scope of our study.

The network architecture consists of an input layer with 1,536 features, followed by a hidden layer of 128 neurons (ReLU activation), another hidden layer with 64 neurons (ReLU activation), and an output layer with 57 neurons (softmax activation) corresponding to the unique CWE classes. We trained the model using the Adam optimizer and sparse categorical cross-entropy loss over 40 epochs with a batch size of 32.

The validation phase was performed by using the weighted F1-score as metric. Based on that, the best checkpoint within the 40 epochs was identified and subsequently employed for inference tasks. This model was assessed on the validation data, revealing that, on average, there were no improvements beyond 23 epochs, with the latest improvement occurring at the 27th epoch. The training parameters were

determined empirically through iterative experimentation and evaluation, as our aim was to test the methodology rather than achieve the optimal model.

4.Inference

After completing training and validation on a balanced dataset, we employed the finalized neural network model to predict the CWE associated with each entry in the unbalanced test set.

8 Evaluation

To evaluate the impact of key terms in predicting CWEs from CVEs, we first assess the effect of including different sections of the CVE descriptions as input to a generic embedding-based model (Section 8.1). This analysis enables us to identify which parts of the CVE have the most significant impact on the model's decision-making process. After identifying the terms that contribute most effectively to classification. In Section 8.2, we evaluate the improvement gained by using a transformer-based architecture over more traditional ones, highlighting the rapid pace of transformer improvements. Section 8.3 examines the influence of the specific information contained within CVE descriptions themselves. Finally, Section 8.4 further explores how these key terms enable stronger differentiation among closely related CWE subclasses.

8.1 Scenario comparison

To evaluate the multiclass classification problem, we used precision, recall, and F1-score based on the weighted average and macro average. Unlike the weighted average, which considers the support (i.e., the number of true instances) for each class, the macro average treats each class equally, regardless of its size or frequency in the dataset.

The paper aims to compare and show which parts of a CVE description contain the most useful information to automate mapping CVEs to their corresponding CWEs. It does not aim to achieve state-of-the-art results in this complex classification but rather improve the performance for a given method as shown in Table 3. The table shows that the CVE representation based on key terms (concatenation of core, contextual, and consequences terms) outperforms all other approaches in both F1-score and accuracy. Specifically, this method outperforms utilizing the complete CVE description, exhibiting an improvement of 8.88% in the F1-score. Other combinations of core terms with contextual or consequence information perform similarly to each other and outperform the scenario using full CVE descriptions.

Subpar results are observed in scenarios where subjects are replaced or omitted. Specifically, substituting the subject with a fixed phrase: "This vulnerability", makes all CVE descriptions appear more similar to one another than they actually are, as they now share the same generic subject. Omitting the subject and substituting it with a blank space makes it more difficult for transformers to use in the embedding process, as it cannot properly relate the verb and the remaining sentence to an unspecified subject. This result highlights the need for a more flexible tool that doesn't rely on fixed text

Table 3: Performance comparison of CWE classification using two distinct approaches: (1) **LLM Prompting**, where a language model is directly prompted with a CVE description, and (2) **Embedding-Based Classification**, where extracted components from the CVE text are embedded and classified using a downstream model. Bold indicates best performance.

		Weighted Avg			Macro Avg		
Approach	**Input format**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
LLM Prompting	Chain-of-thought	69.06%	58.25%	60.52%	11.77%	9.80%	9.83%
	Zero-shot	71.96%	61.03%	62.27%	10.74%	8.76%	8.72%
Embedding Based	Full CVE description	71.18%	63.58%	64.62%	51.03%	62.89%	54.66%
	Without subject	66.01%	54.96%	56.16%	46.33%	52.93%	47.44%
	Replaced subject	67.65%	58.57%	60.28%	47.42%	53.54%	48.51%
	Key terms	**74.07%**	**69.84%**	**70.36%**	57.52%	**64.85%**	**59.66%**
	Core terms	72.61%	69.17%	69.48%	**57.97%**	63.62%	59.39%
	Core + context	73.71%	68.35%	69.40%	55.70%	64.25%	58.09%
	Core + consequences	72.75%	68.07%	68.27%	55.94%	64.33%	58.24%

processing rules. In some CVE descriptions, the root cause and vulnerability type are mentioned at the beginning of the description, requiring a more adaptable approach.

By conducting a detailed analysis of CVE descriptions and key terms, we evaluate the effectiveness of our approach in improving performance across various CWE classes. On the macro average, there is a consistent improvement in the performance across all configurations compared to using only the CVE description. However, the low scores highlight the inherent difficulty of the task, regardless of the number of instances per class. For example, CWE-312 (Cleartext Storage of Sensitive Information) is frequently misclassified as CWE-200 (Exposure of Sensitive Information to an Unauthorized Actor) 33.33% of the time. Although CWE-312 has only 42 instances, these misclassifications significantly impact the model's performance on the macro average.

On the other hand, the enhancement observed in classes with higher frequencies within the original dataset was particularly notable. Specifically, CWE-79 (Cross-site Scripting) and CWE-89 (SQL Injection) together account for over 15% of the entire dataset, with observed improvements on the weighted F1-score of 10.43% and 10.95%, respectively.

To establish a strong baseline, we compared our method to LLM-only approaches by querying GPT via an API for CWE classification. Specifically, we reproduced the chain-of-thought prompting strategy proposed by Liu et al. [15], adopting their best-performing prompt configuration. In addition, we evaluated a zero-shot setting, where GPT was prompted with a CVE description and tasked with predicting the most appropriate CWE label without reasoning steps or examples. Both prompting strategies underperformed relative to our embedding-based methods. While the weighted-average scores of the LLM-based approaches are comparable, their macro-average performance is significantly lower. For a controlled comparison, we constrained GPT's output to the same set of 57 CWE classes used for training and evaluation in our method. Even under this restriction, LLM-only approaches failed to match the performance of the embedding-based classifiers.

8.2 Transformer-based vs traditional architectures

Our evaluation leverages transformer-based embeddings because of their well-established ability to capture sentence-level semantics and contextual relationships, as demonstrated in [7, 18]. In this work, we not only adopt these modern techniques but also extend our investigation to more traditional architectures, thereby demonstrating that our preprocessing method based on key term extraction is robust and generalizable across a range of models and learning paradigms.

Table 4: Comparison of CWE classification performance using the CVE full-description vs. term-extracted inputs across four scenarios: (1) Unsupervised Transformer-based model [18], (2) Supervised Transformer-based model (ours), (3) Supervised TextCNN [17], and (4) Supervised BiGRU-TextCNN [17]. Each model is evaluated on both the full CVE description ("Desc.") and extracted key terms ("Terms"). All results are reported as weighted averages. Bolded values indicate performance improvements from term-based input.

Method	Architecture	CVE	Prec.	Rec.	F1	Acc.
1) Unsupervised	Transformer-based	Desc.	53.53%	33.03%	31.94%	33.03%
		Terms	**54.39%**	**39.31%**	**37.84%**	**39.31%**
2) Supervised	Transformer-based	Desc.	71.18%	63.58 %	64.62%	63.58 %
		Terms	**74.07%**	**69.84%**	**70.36%**	**69.84%**
3) Supervised	TextCNN	Desc.	**75.32%**	68.96%	70.09%	68.96%
		Terms	75.20%	**70.39%**	**71.34%**	**70.39%**
4) Supervised	BiGRU-TextCNN	Desc.	**75.52%**	70.16%	70.59%	70.16%
		Terms	75.11%	**71.30%**	**71.64%**	**71.30%**

Table 4 summarizes the performance obtained with both unsupervised and supervised approaches. In the transformer-based setting, the use of key terms instead of the full CVE descriptions yields an improvement consistent with the supervised learning (e.g., a 5.9 percentage point increase in the F1-score). Similar trends emerge with supervised models based on more traditional architectures, including TextCNN and BiGRU-TextCNN, although the improvements from key term representations are comparatively smaller. This observation reinforces the hypothesis that key terms provide a better representation for CVE-to-CWE mapping than the complete, and often noisy, descriptions.

The more modest improvements seen in the TextCNN and BiGRU-TextCNN architectures can be attributed to their dependence on capturing sequential and local context. When the CVE descriptions are reduced to only their essential key terms, some of the detailed sequential information is necessarily lost, which limits the performance gains in architectures optimized for such context. Nevertheless, the fact that key term extraction yields improvements across both unsupervised and supervised learning paradigms highlights its utility as a consistent enhancement, regardless of the underlying learning framework.

Furthermore, we investigated the generalizability of our approach across a range of transformer-based models representing an evolutionary timeline, from early bidirec-

tional models like BERT [8] to more recent generative architectures such as Llama [22] and Phi [14], and finally the state-of-the-art, GPT. In our experiments, embeddings were obtained using different strategies for each model (e.g., averaging all token embeddings in BERT versus using the last token in Phi and LLama). Despite these extraction differences, as illustrated in Figure 4 and reported in Appendix 12.5, the rapid and consistent improvements associated with key term representations remain evident. Importantly, given the rapid evolution of transformer-based models and the

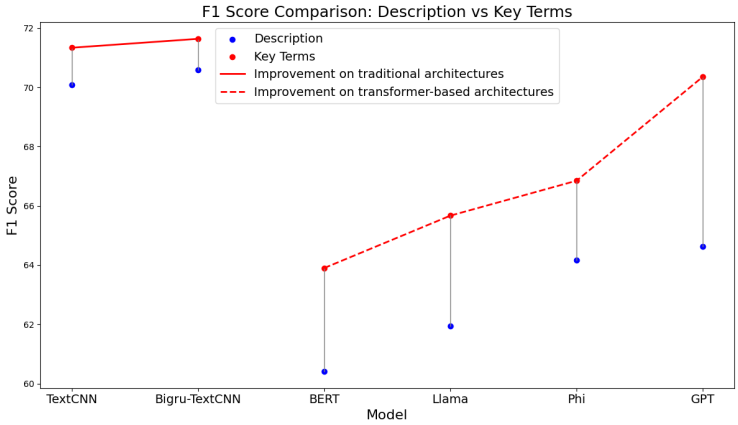

Fig. 4: Improved CVE-to-CWE classification by leveraging key terms instead of full-text descriptions across different architectures.

consistent gains observed with key term extraction, we anticipate that the combination of transformer architectures with key term extraction will soon outperform more traditional methods. This trend is evident across both unsupervised and supervised learning paradigms, reinforcing our central thesis: focusing on the core semantic elements within CVE descriptions not only enhances current mapping performance but also forecasts even greater improvements as transformer models continue to advance.

This consistent improvement, along with the broad generalizability of the approach, underscores the fundamental insight that isolating the key semantic information in CVE descriptions is a robust strategy for boosting CVE-to-CWE mapping performance across diverse modeling and learning frameworks.

8.3 Long vs short CVE descriptions

We explore whether the improvements achieved through key term extraction vary based on the length of CVE descriptions. By leveraging the extracted key terms, we observe an overall performance improvement across all data lengths, with particularly significant enhancements for both long and short CVE descriptions, as shown in Table 5.

The key terms extraction process effectively identifies and emphasizes the crucial elements of the description required for predicting the corresponding CWE. As shown in

Table 5, the term extraction captures the essential information in lengthy CVE descriptions, while for shorter descriptions it reinforces the fundamental concepts. Essential terms such as 'Cross-site Scripting', 'XSS', and 'Web Security' are identified and appropriately replicated as required. Notably, in shorter descriptions, we observe the effective utilization of the inherent knowledge within the Language Model because it recognizes that XSS (Cross-Site Scripting) is related to web security and is a web vulnerability.

Table 5: Examples of Long and Short CVE Descriptions with Extracted Terms

Long CVE: A vulnerability was found in SourceCodester Class Scheduling System 1.0. It has been declared as critical. Affected by this vulnerability is an unknown functionality of the file `/admin/edit_subject.php` of the component GET Parameter Handler. The manipulation of the argument id leads to sql injection. The attack can be launched remotely. The exploit has been disclosed to the public and may be used. The identifier VDB-229597 was assigned to this vulnerability.

Core Terms	**Contextual Terms**	**Consequences**
SQL Injection	SourceCodester Class Scheduling System	Critical vulnerability
Remote Attack	GET Parameter Handler	Remote exploitation
Public Disclosure	File Manipulation	Public disclosure

Labels: Real: 89, Predicted using description: 94, Predicted using terms: 89

Short CVE: kkcms 1.3 has `jx.php?url=` XSS.

Core Terms	**Contextual Terms**	**Consequences**
Cross-site Scripting	kkcms 1.3	Data manipulation
XSS	jx.php?url=	Untrusted input
Web Security	URL parameter	Web vulnerability

Labels: Real: 79, Predicted using description: 611, Predicted using terms: 79

To generalize the improvements observed for both long and short CVE descriptions using our method and to ensure robust and reliable results, we restricted the CVE descriptions to those containing between 9 and 69 words. This range was chosen to minimize excessive variability in the analysis, as relying on descriptions with too few samples could skew the results and distort the graphical representation. The word count distribution is notably uneven, resembling a Gaussian curve with a pronounced right tail compared to the left one. The graphical representation in Fig. 5 depicts the misclassification percentage (left y-axis), indicating the frequency of misclassifications by the model, with lower values signifying better performance. The solid line denotes the average misclassification, with the green line corresponding to the performance of the key terms approach, exhibiting a lower misclassification than the comprehensive descriptions. The dashed lines represent the moving average of predictions related to the length of sentences. Especially, the green line consistently outperforms the descriptions' misclassification rate. The average misclassification demonstrates a consistent improvement of our approach across all CVE descriptions. Specifically, the key terms approach demonstrates an average misclassification rate of 30.16%, compared to 38.58% for the

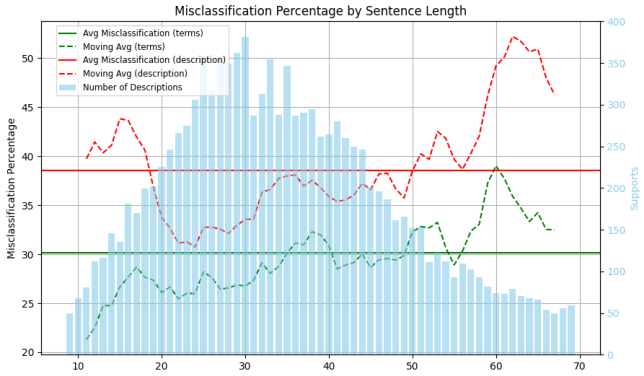

Fig. 5: Misclassification percentages by sentence length, showing consistent improvement of the key terms approach over comprehensive descriptions.

CVE descriptions, reflecting a consistent improvement of over 8 percentage points. By averaging the misclassification rates for CVE descriptions longer than 64 words (specifically the last five, ranging from 65 to 69 words), our method shows an improvement of 13.68 percentage points. Similarly, for CVE descriptions shorter than 14 words (specifically the first five, ranging from 9 to 13 words), the improvement is 18.32 percentage points. These results highlight a consistent enhancement across all data lengths, with particularly significant performance gains in handling extreme cases of very long and concise descriptions. To give more context, the blue bars represent the distribution of CVE descriptions by word count, with at least 50 elements to be included.

8.4 Subtle differentiation of CWE Siblings

The primary advantage of term extraction is its capability to discern subtle distinctions among CWE siblings (children of the same parent). This can be seen when we compare the results of key terms vs. CVE descriptions to identify the CWE *classes* (CWE parent) related to all the CWE labels as shown in Table 6. The variation in this classification is less pronounced than that observed in the CWE base classification.

Table 6: Model performance of mapping CVE descriptions to CWEs classes (parent) on the weighted average. **Note**: This table differs from Table 3 because, instead of classifying CVE descriptions in CWE base (57 classes), in this case we classify CVE descriptions in CWE *classes* (28 classes).

CVE	**Precision**	**Recall**	**F1-score**	**Accuracy**
Description	79.10%	76.52 %	76.87%	76.52%
Key terms	80.39%	79.56%	79.48%	79.56%

To showcase term extraction from closely related CWE siblings, consider the following example involving a buffer overflow vulnerability:

> "This vulnerability allows remote attackers to execute arbitrary code on affected installations of OpenText Brava! Desktop 16.6.3.84. User interaction is required ... which can result in a write past the end of an allocated buffer... "

The vulnerability involves a buffer overflow, and both the description and extracted terms can recognize it. However, the distinguishing factor lies in whether the buffer overflow occurs during the writing or reading process. While the classification based on the CVE description incorrectly labels the entry as a 'Buffer-Overflow read', the extracted terms accurately clarify that the vulnerability pertains to the writing process 'Buffer-Overflow write', as explicitly stated in the contextual term 'Write past end of buffer'.

9 Discussion and future work

Our study successfully distilled key information from CVE descriptions, highlighting the importance of focusing on essential elements while filtering out irrelevant details for effective CVE-to-CWE mapping. We observed that key term representations consistently outperform alternative methods, underscoring a critical challenge: CVE descriptions are often verbose and contain irrelevant information for CWE classification. While this additional information is valuable for other applications, automated CVE-to-CWE mapping methods must account for and possibly mitigate the impact of such noise.

A critical challenge in extracting key terms using LLMs is their tendency to generate hallucinations. In Sec. 6.2, we evaluated the models while considering these hallucinations as a key factor in the model selection process. Although GPT exhibits a relatively low rate of hallucinations, we further analyzed the seven extracted terms where hallucinations were detected during manual evaluation. Our findings indicate that four of these instances were correctly classified under the appropriate CWE, one was assigned to the correct CWE *class* (parent) but to the incorrect CWE base (child), and two were misclassified. Despite the small sample size, the overall CWE prediction accuracy for key terms with hallucinations was 57.14%, compared to the original model's accuracy of 69.84% (see Table 3). When the classification constraint is relaxed to CWE parent categories, the accuracy increases to 79.56% for the original model (see Table 6) and 71.43% for hallucination-affected cases. Although this represents a drop in performance, the impact of hallucinations is diluted, as each hallucination affects only a single term within a set of nine extracted terms (3 terms for each core, contextual, and consequences). Future research will explore to which extent we can reduce or correct hallucinations by using techniques such as self-verification [25].

We selected GPT for key term extraction over Llama due to its better performance. However, the proprietary nature of GPT imposes cost and accessibility constraints, particularly for newer models. An alternative is fine-tuning an open-source LLM, such as Phi or LLaMA, for key term extraction. This approach reduces costs and enhances data privacy, but, as discussed in Section 6.2, comes at the expense of performance, as GPT has demonstrated better extraction capabilities compared to LLaMA on the base models.

10 Limitations

The technique presented in this study does not encompass all possible CWEs, as it has not been exhaustively tested across the entire CWE spectrum. NIST recognizes a subset of 130 CWEs (CWE-1003 view), including classes, bases, and variants. Since we did not test our method across all possible CWEs, we cannot claim that our approach is generalizable to the entire CWE spectrum. However, our goal in this paper is not to classify CVEs into as many CWEs as possible but to demonstrate the effectiveness of key term extraction in this context.

A key limitation of using closed-source LLMs like GPT-3.5 Turbo lies in the cost per request. Appendix 12.4 provides a detailed comparison of the per-request costs for various state-of-the-art LLMs, both open and proprietary. For open-source models, costs stem primarily from infrastructure usage. Among the models considered, GPT-4.1-nano emerged as the most cost-efficient, while Claude Sonnet 4 and Grok 4 were the most expensive. Notably, even under the most expensive configuration, replicating the experiments presented in this study would result in a total cost of approximately 200.

Although achieving state-of-the-art performance in this classification task was not our primary goal, we still enhanced all architectures, especially transformer-based ones. Instead, our focus is on determining which parts of a CVE description offer the most valuable information for automating CVE-to-CWE mapping.

11 Conclusions

This study introduces a methodology aimed at enhancing the automation of mapping CVEs to their corresponding CWEs. By leveraging in-context learning with LLMs, we extract key terms from CVEs that are highly relevant to CWE identification, enhancing the F1-score and accuracy of the mapping processes.

The comparative analysis of different CVE representation methods highlights the improvements brought by the term-based approaches in capturing the semantic essence of CVEs. Additionally, the study explains the impact of description length on key term extraction performance, showcasing the robustness of the methodology across various lengths of CVE descriptions.

Furthermore, this study demonstrates the embedding generalizability to multiple language models, showcasing its potential across language models of different sizes. The methodology also improves traditional CVE-to-CWE classification methods, although the gains are more modest compared to those observed with transformer architectures.

12 Appendix

12.1 Mapping CVEs to CWEs over time

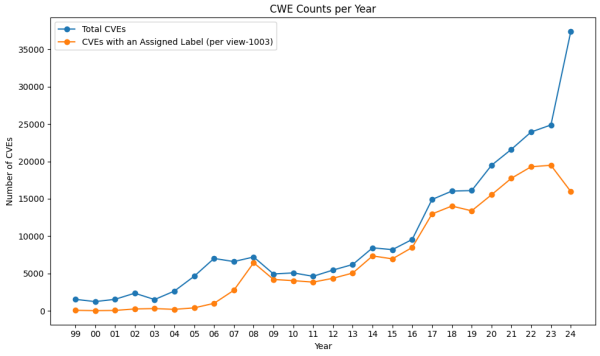

Fig. 6: The graph displays two trends: the total number of CVEs reported each year and the subset of CVEs that have an assigned a CWE label according to the 1003-view.

12.2 Prompt Design

To extract key terms from each CVE description, we designed a structured prompt for the language model. The CVE description was inserted dynamically in the prompt using the placeholder **cve_description**. The prompt aimed to guide the model in generating a well-organized and semantically relevant keyword list by providing task instructions, keyword generation tips, and a worked example:

> I want you to create a proper keyword list for a CVE, I will give you a list of tips for proper keyword list creation which are given below between triple backticks. Then I will give you and example CVE and the example keyword list in JSON, to help you understand the task better. Your result should also have 3 related keywords per item like the example result. Lastly I will give you a CVE and you have to give me the keyword list in JSON format.
> Tips:
> 1. Text Normalization: Standardize the text by converting it to lowercase, removing special characters, and expanding abbreviations. This helps in reducing variations in the text.
>
> 2. Abstraction: given the CVE description, create an abstraction of it by leaving aside the terms that are too specific (e.g. if a CVE is specifically targeting a cisco router, this detail is not important)

3. Keyword Extraction: Identify and extract key terms and concepts from both the CVE and CWE descriptions. For instance, phrases like "incorrectly handles a length field" or "buffer overflow" are critical.

CVE_example:
"CVE description": "Patient Information Center iX (PICiX) Versions B.02, PerformanceBridge Focal Point Version A.01, IntelliVue patient monitors MX100 IntelliVue X3 and X2 Versions N and prior. The software parses a formatted message or structure but does not handle or incorrectly handles a length field that is inconsistent with the actual length of the associated data, causing the application on the surveillance station to restart."
CVE_example result:
"core terms": ["Parsing Vulnerability", "Inconsistent Length Field Handling", "Application Restart"],
"contextual terms": ["Length field", "Inconsistency", "Improper handling"],
"consequences": ["Unexpected restart", "Data length mismatch", "Application instability"]

CVE: "**cve_description**"
Give the result like the example result in a JSON list, with proper representation of the keywords list applicable for this CVE.

12.3 Hallucination, Completeness, and Consistency: Illustrative Examples

In this section, we provide concrete examples of CVE descriptions that demonstrate the presence or absence of the three key attributes when using large language models for term extraction: **hallucination**, **incompleteness**, and **inconsistency**.

1. Hallucination Example: The LLM introduces concepts not grounded in the source text.

- **CVE Description:**
  "IBM Security Directory Suite VA 8.0.1 through 8.0.1.19 stores user credentials in plain clear text which can be read by an authenticated user. IBM X-Force ID: 228567."
- **Extracted Core Terms:**
  - Credentials Exposure
  - Clear Text Storage
  - **Authentication Bypass** *(hallucinated)*
- **Explanation:** The CVE explicitly states that the credentials are accessible *by an authenticated user*, not that authentication can be bypassed. The inclusion of "Authentication Bypass" is a clear hallucination.

2. Incompleteness Example: Key information from the CVE is missing in extracted terms.

- **CVE Description:**
  "An improper neutralization of special elements used in an OS command vulnerability

(CWE-78) in FortiADC 7.2.0, 7.1.0 through 7.1.1 may allow an authenticated attacker to execute unauthorized commands via specifically crafted arguments to existing commands."

- **Extracted Terms (GPT):**
  - **Core Terms:** Command Injection, OS Command Vulnerability, Authenticated Attack
  - **Contextual Terms:** Crafted Arguments, Unauthorized Commands, Specifically Crafted
  - **Consequences:** Unauthorized Code Execution, System Compromise, Authenticated Attack
- **Explanation:** The extraction fails to include "improper neutralization of special elements," which is a critical technical detail indicating the root cause (CWE-78). Instead, it focuses only on the effect and context.

3. Inconsistency Example: Repeated extractions from the same CVE yield differing concepts.

- **CVE Description:**
  "The Zoom Client for Meetings chat functionality was susceptible to Zip bombing attacks in the following product versions: Android before version 5.8.6, iOS before version 5.9.0, Linux before version 5.8.6, macOS before version 5.7.3, and Windows before version 5.6.3. This could lead to availability issues on the client host by exhausting system resources."
- **Extracted Core Terms (Three Runs):**
  1. Zip Bombing, Availability Issues, Resource Exhaustion
  2. Zip Bombing Vulnerability, Chat Functionality Vulnerability, Resource Exhaustion
  3. Zip Bomb Vulnerability, Chat Functionality Vulnerability, Resource Exhaustion
- **Explanation:** While runs 2 and 3 yield consistent outputs, run 1 omits "Chat Functionality Vulnerability" and instead includes a broader effect term ("Availability Issues"). This variability illustrates inconsistency in GPT's term extraction process.

12.4 Terms Extraction Cost

Key term extraction can be performed using either open-source or closed-source LLMs. This study explores both options by selecting the best-performing models publicly available at the time of writing. For the open-source implementation, we deployed **LLaMA 2 (7B)** on an A40 GPU, with the 7B parameter size chosen because of hardware limitations. For the closed-source implementation, we accessed **GPT-3.5 Turbo** via API.

To contextualize the cost of key term extraction, we compare the actual costs incurred in this study with estimated costs of executing the same task using a range of current state-of-the-art LLMs, both open and proprietary. Additionally, we provide an estimation of the total cost required to process the entire CVE dataset, which as of August 2025 includes 293,623 entries.

It is important to note that these costs are incurred only once during the extraction process and are not required for every subsequent training or evaluation of the downstream model. Table 7 summarizes the cost for both the experimental subset used in this study and the projected cost for the full dataset. While runtime pricing is reported for transparency, it is worth noting that most providers offer a discounted rate (approximately 50%) for batch processing via asynchronous APIs, which would be applicable in our case despite longer processing times.

Table 7: Cost of extracting key terms for this study and for the entire CVE dataset (293,623 CVEs as of August 2025) using different LLMs.

Scenario	GPT-4.1	GPT-4.1 Mini	GPT-4.1 Nano	GPT-3.5 Turbo	Claude Sonnet 4	DeepSeek	Grok4
This paper	$123.74	$24.75	$6.19	$28.40	$200.84	$33.98	$200.84
Whole CVE dataset	$426.37	$85.27	$21.32	$97.85	$692.04	$117.08	$692.04

12.5 Embedding comparison

Table 8: F1 performance and gain from using Key Terms instead of full CVE description as input, across different language models used as embedder for the downstream classification.

Model	Input Type	Prec.	Rec.	F1	Relative F1 Gain
BERT	Description	67.79%	59.10%	60.41%	**+5.78%**
	Key Terms	**69.23%**	**63.50%**	**63.90%**	
Phi	Description	71.37%	63.15%	64.17%	**+4.18%**
	Key Terms	**70.02%**	**66.55%**	**66.85%**	
LLaMA	Description	72.50%	60.58%	61.94%	**+6.57%**
	Key Terms	**73.29%**	**64.87%**	**66.01%**	
GPT	Description	71.18%	63.58%	64.62%	**+8.88%**
	Key Terms	**74.07%**	**69.84%**	**70.36%**	

References

1. Nvd data feeds. `https://nvd.nist.gov/vuln/data-feeds`, accessed on: 23-1-2024
2. Aghaei, E., Shadid, W., Al-Shaer, E.: Threatzoom: Hierarchical neural network for cves to cwes classification. In: International Conference on Security and Privacy in Communication Systems. pp. 23–41. Springer (2020)
3. Aota, M., Kanehara, H., Kubo, M., Murata, N., Sun, B., Takahashi, T.: Automation of vulnerability classification from its description using machine learning. In: 2020 IEEE Symposium on Computers and Communications (ISCC). pp. 1–7 (2020). `https://doi.org/10.1109/ISCC50000.2020.9219568`
4. Bafna, P., Pramod, D., Vaidya, A.: Document clustering: Tf-idf approach. In: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). pp. 61–66. IEEE (2016)
5. Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., Jatowt, A.: Yake! keyword extraction from single documents using multiple local features. Information Sciences **509**, 257–289 (2020)
6. CVE Project: Cve project documentation. `https://cveproject.github.io/docs/` (2023), accessed: 2024-09-04
7. Das, e.a.: V2w-bert: A framework for effective hierarchical multiclass classification of software vulnerabilities. In: 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA). pp. 1–12. IEEE (2021)

8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019)
9. Dherin, B., Munn, M., Mazzawi, H., Wunder, M., Gonzalvo, J.: Learning without training: The implicit dynamics of in-context learning. arXiv preprint arXiv:2507.16003 (2025)
10. Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Liu, T., et al.: A survey on in-context learning. arXiv preprint arXiv:2301.00234 (2022)
11. Grootendorst, M.: Keybert: Minimal keyword extraction with bert. (2020). `https://doi.org/10.5281/zenodo.4461265`, `https://doi.org/10.5281/zenodo.4461265`
12. Han, Z., Li, X., Liu, H., Xing, Z., Feng, Z.: Deepweak: Reasoning common software weaknesses via knowledge graph embedding. In: 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER). pp. 456–466. IEEE (2018)
13. Hutchins, E.M., Cloppert, M.J., Amin, R.M., et al.: Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Leading Issues in Information Warfare & Security Research **1**(1), 80 (2011)
14. Javaheripi, M., Bubeck, S.: Phi-2: The surprising power of small language models. Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL) **2**(1) (December 2023), published version
15. Liu, X., Tan, Y., Xiao, Z., Zhuge, J., Zhou, R.: Not the end of story: An evaluation of chatgpt-driven vulnerability description mappings. In: Findings of the Association for Computational Linguistics: ACL 2023. pp. 3724–3731 (2023)
16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
17. Pan, M., Wu, P., Zou, Y., Ruan, C., Zhang, T.: An automatic vulnerability classification framework based on bigru-textcnn. Procedia Computer Science **222**, 377–386 (2023)
18. Simonetto, S., van Ede, T.S., Bosch, P., Jonker, W.: Text2weak: mapping cves to cwes using description embeddings analysis. In: 4th Workshop on Artificial Intelligence-Enabled Cybersecurity Analytics (2024)
19. Simonetto, S., Oostveen, R., van Ede, T.S., Bosch, P., Jonker, W.: Knowing your weaknesses is your greatest strength: Mapping cve to cwe by leveraging cwe hierarchy and llms (2025)
20. Sun, J., et al.: Aspect-level information discrepancies across heterogeneous vulnerability reports: Severity, types and detection methods. ACM Transactions on Software Engineering and Methodology **33**(2), 1–38 (2023)
21. Sun, X., Ye, Z., Bo, L., Wu, X., Wei, Y., Zhang, T., Li, B.: Automatic software vulnerability assessment by extracting vulnerability elements. Journal of Systems and Software **204**, 111790 (2023)
22. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models (2023), `https://arxiv.org/abs/2302.13971`
23. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
24. Wang, Q., Gao, Y., Ren, J., Zhang, B.: An automatic classification algorithm for software vulnerability based on weighted word vector and fusion neural network. Computers & Security **126**, 103070 (2023)
25. Weng, Y., Zhu, M., Xia, F., Li, B., He, S., Liu, S., Sun, B., Liu, K., Zhao, J.: Large language models are better reasoners with self-verification. arXiv preprint arXiv:2212.09561 (2022)