From Descriptions to Decisions: Classifying Vulnerabilities by Information Sufficiency

Bejruz Domínguez Adilova
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
b.dominguezadilova@student.utwente.nl

***Abstract*—In cybersecurity, Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration (CWE) are the industry standard for registering a vulnerability and categorizing a weakness, respectively. About 30% of CVEs are not labeled with a CWE and approximately 50% of that subset is unlabeled due to poor descriptions. A significant portion of CVEs are not labeled with all their relevant CWEs. The existing standards for CVE descriptions are generally not adhered, causing issues for attempts to automate their CWE labeling. A binary classifier could potentially flag CVE entries with insufficient description information to properly label it to a CWE. This work proposes exploring different binary classifiers utilizing Natural Language Processing (NLP) models in order to assess the viability of an automated classifier for determining the labeling sufficiency of a vulnerability description given the available CVE datasets. BERT and Random Forest, two vastly different models, based on transformers and decision trees respectively, provided similarly promising results, between 83% and 87% accuracy in their best performing models.**

***Index Terms*—CVE, CWE, NLP, BERT, binary classification, random forest**

I. Introduction

Common Vulnerabilities and Exposures, or CVEs, are unique identifiers for publicly disclosed cybersecurity vulnerabilities [1]. These identifiers are stored in conjunction with a relevant description, provided by the registering party. The descriptions can be of vastly varying quality and completeness, as the existing standards are not fully adhered to [2]. This system has been in use since 1999 and has been the industry standard ever since, enabling fully new services such as the NIST's U.S. National Vulnerability Database [3].

Common Weakness Enumeration, or CWE, is an agglomeration of common weakness types, in terms of both software and hardware, that can lead to exploitable vulnerabilities [4]. The MITRE Corporation, the same entity behind the inception of CVEs in 1999, is responsible for creating the CWE list in 2006. All CVEs can be labeled with a corresponding CWE to group them by the type of weakness to be exposed [5]. Despite the different coding practices of developers from varied backgrounds, the CWE List enables them to share the same type of common weaknesses by offering a standardized format. In other words, labeling a CVE with an accurate CWE can aid security researchers in eliminating or mitigating security weaknesses [4]. Multiple CWEs can be given to a CVE, depending on the category, as some CWE are higher level, whilst others are more specific. Alternatively, a vulnerability can exploit more than one weakness, also leading to multiple CWEs labels.

A CVE does not require to be labeled manually by the author with an adequate CWE during its registration; only a non-standardized description of the CVE is required. Considering the added value by having a CWE category attached to a CVE, there has been extensive research attempting to automate the CVE to CWE labeling process. There is a plethora of approaches. However, a common point is that all of them rely on the CVE description as the main data input for their algorithm of choice [2] [6] [7] [8] [9] [10] [11]. Depending on the methodologies, dataset, and granularity, the accuracy can range anywhere between 70%, for implementations taking into account 130 CWE classes (MITRE View-1003) [10], to 97%, for implementations with 124 CWE classes distributed across a three layer hierarchy. [6].

The datasets available are polluted and contain CVE entries with CWE labels that should be, based on the descriptions available, unlabeled. Further, the datasets can suppose additional challenges given the nature of the descriptions, with overly specific or technical terms.

Given this context, it can be surmised that the industry could benefit from having better, in other words, more extensive and relevant, descriptions attached to CVEs in order to improve the accuracy of the automated systems linking them to relevant CWEs, ultimately allowing for improved security research. Some entries provide little useful information by providing generic descriptions, whilst others provides clear technical insights [2]. Both are exemplified in Table I.

The quality of descriptions in CVE datasets could be improved via two methods. First, the registerer of a new CVE entry could be given instant feedback on the lack of their description to avoid having to retroactively fix it later. Second, filtering large dataset of CVE entries to remove the ones deemed to have insufficient descriptions. This could enhance the accuracy of the aforementioned automated CWE labeling algorithms. Binary classifiers using natural language processing techniques could be used to this end [12] [13] [14].

In this research, five binary classifiers are implemented, extensively trained, and tested to determine whether a CVE's description is sufficient to be labeled with CWE categories.

TABLE I: CVE examples

CVE	Description	Quality
CVE-2015-9067	In all Qualcomm products with Android releases from CAF using the Linux kernel, a potential compiler optimization of memset() is addressed.	Inadequate, no CWE labels.
CVE-2017-17067	Splunk Web in Splunk Enterprise 7.0.x before 7.0.0.1, 6.6.x before 6.6.3.2, 6.5.x before 6.5.6, 6.4.x before 6.4.9, and 6.3.x before 6.3.12, when the SAML authType is enabled, mishandles SAML, which allows remote attackers to bypass intended access restrictions or conduct impersonation attacks.	Adequate, contains CWE label.

II. Problem statement

There has been ample research done on both NLP-based binary classifiers, as well as, automating CVE to CWE labeling, independently. However, there is little to no research on the use of one to enhance the other, meaning the use of binary classifiers in the field of CVE labeling. This paper will delve into different NLP based methods for binary classification and attempt to find the most suitable implementation for the needs of the research questions stated below.

A. Research Question

How reliably can a binary classifier determine whether a CVE description contains enough information to be mapped to an accurate CWE category?

The following questions can be used to help answer the main Research Question:

1) Sub-RQ1: How do different Machine Learning models perform on the CVE datasets?

2) Sub-RQ2: To what extent is the available CVE dataset suitable to machine learning tasks?

3) Sub-RQ3: How well can a model perform on synthetic samples unrelated to cybersecurity?

III. Related Work

The search for related works has been conducted in two rounds. The first round consisted of exploring research conducted on binary classifiers that used NLP methods, in order to gauge where they stand in terms of complexity, explainability, and accuracy. The second round delved into recent literature on attempts to automate CVE labeling with CWEs.

A. Classifiers

There is a plethora of classification methods that offer different results. Udin et al. [13] showed promising numbers, namely a 94% accuracy, in their 2022 paper when using the 'random forest' method for binary classification on a drug review dataset. This method has a moderate complexity as well as interpretability. XGBoost was selected as a more sophisticated decision tree based model that could theoretically offer higher accuracy due to boosting and regularization.

Two bi-directional transformer based models were also chosen, 'BERT' and 'XLNet', the latter of which offers a 96% accuracy, as stated by Arabadzhieva et al. [12] in a 2022 paper, assessing a dataset of 50,000 reviews in English. Furthermore, 'CNN' was also chosen based on the example by Bao et al, where they achieved a 88.53% accuracy when classifying medical literature [15].

The aforementioned methods were selected due to all of them having a high reported or expected accuracy level, whilst different in the other relevant aspects, namely complexity and computation, as well as technical implementation.

B. CVE to CWE labeling

Most of the works explored on the subject of labeling automation outlined how CVE descriptions were a potential limiting factor [11] [9] [10] [6]. According to Oostveen's 2024 paper [9], CVE descriptions are frequently incomplete, leading to labeling inconsistencies, by, for instance, assigning a single CWE to a vulnerability to which multiple CWE categories applied. In their 2024 paper, Sun et al. [11] argued that the descriptions could be short and full of technical jargon, making algorithms more prone to mislabeling them due to focus on irrelevant tokens. In the same year, Kota et al. [10] plainly state that many CVEs remain unlabeled due to the lack of detail in their descriptions to assign them a CWE label with certainty. Finally, Das et al. [6] identified in their 2021 paper that the gap in the CVE descriptions and CWE definitions could make fine-grained CWE categories harder to get assigned to. What all these papers have in common is the they point at the CVE descriptions as a shortcoming to be overcome.

Notably, no research has been conducted on applying the previously stated classifiers on CVE descriptions, making it a potential area of interest for exploration.

IV. Methodology

Considering how this research involves exploring aspects of accuracy with binary classification methods, experimentation with numerous implementations is be required. First, Random Forest, XGBoost, BERT, CNN and XLNet will be explored with the CVE dataset. Exploring plurality of options allows for an educated answer to be developed for the main research question.

A. Data selection and processing

A CVE dataset was obtained directly from the MITRE organization, organized in `json` files. There is a file for each year from 1999 to 2024, containing all the CVEs registered in that year. Each entry contains a CVE id, its description, and its respective CWE labels. For entries without a known CWE tag, a `No-Info` label is attached in the field.

The dataset is rather polluted. Some entries contain the `No-Info` label in the CWE column, whilst also offering an actual CWE tag, which is conflicting. In these cases, it is assumed that the description is not sufficiently rich to assign a CWE tag, but a relevant entity with more background information assigned the corresponding tag. For the purposes of this research, these conflicting entries were treated the same as all the other `No-Info` entries. An example of both a

regular `No-Info` entry mentioned prior, as well as this edge case can be seen in Table II.

TABLE II: Sample JSON file CVE entry

CVE ID	Description	CWE
CVE-2003-0789	mod_cgid in Apache before 2.0.48, when using a threaded MPM, does not properly handle CGI redirect paths, which could cause Apache to send the output of a CGI program to the wrong client.	NVD-CWE-noinfo
CVE-2003-1422	Multiple unspecified vulnerabilities in the installer for SYSLINUX 2.01, when running setuid root, allow local users to gain privileges via unknown vectors.	NVD-CWE-noinfo; CWE-16

For the sake of having a standardized dataset for training different models, all the data files were unified into one large `csv` file. Consequently, all the CWE labels were encoded into either a `0` for `No-Info` or `1` for entries with CWE tags. The file was, thereafter, processed to remove duplicate entries. Approximately 9000 entries were removed, about 3000 `No-Info` entries, and about 6000 labeled ones.

At this stage, the resulting `csv` had approximately 220000 labeled entries, and about 28000 `No-Info` entries, meaning only 11% of entries were unlabeled. Considering this heavy imbalance, two new subsets were made. Both subsets contain the full number of `No-Info` entries and randomly select a matching amount of labeled entries, as well as a x2 ratio of labeled entries. Ultimately, 3 datasets were chosen to train models with - 1:1 ratio subset, 1:2 ratio subset, and the full dataset. Refer to Table III for more clarification.

TABLE III: Datasets

Set	Entries	Ratio
Subset 1	53992	1:1
Subset 2	80988	1:2
Full dataset	248824	1:9.2

In order to make the descriptions more uniform for the machine learning tasks at hand, the descriptions were preprocessed with basic steps. These steps include: removing markup (HTML tags), links, non-standard characters, and whitespace, as well as lower-casing the entire text. In the manual testing scripts for the models, which will be explained later on, the same text preprocessing steps are used, to ensure uniformity across results.

Lastly, depending on the model, which is clarified in their respective sections, the dataset is further split into subsets for training and testing, in a 80/20 ratio, for most models except CNN and BERT, which used a training, testing, and validation sets in a 70/15/15 ratio.

B. Model selection

The objective of this research is to use machine learning for natural language processing. Textual descriptions have to be processed to make inferences. In order to get a deeper understanding of what might suit the given dataset, a variety of models was chosen, namely XGBoost, XLNet, BERT, CNN, and Random Forest.

Random forest and XGBoost were chosen as the more basic decision tree-based options, XLNet and BERT as the bidirectional transformer-based models, and lastly CNN as the in between, being more sophisticated at finding patterns though its neural network than the decision tree-based models, but lighter than transformers.

C. Model implementation

All models train separately on each of the established datasets. Each training run performs a grid search trying to optimize parameters relevant to that specific model, before selecting and saving the best one. The trained model is exported for future testing and all relevant metadata such as testing accuracy is saved for reference and comparison. Except for BERT and CNN, an 80/20 train/test split is used.

1) Random Forest: The `Scikit-learn` python modules were used to implement Random Forest. A `TF-IDF` vectorizer is used to extract features from the text, with a hard limit of 2000 features. For the grid search, the number of estimators, as in, the number of trees in the forest, could be either 100 or 200. The maximum depth each tree could go was set to either 10 or 20. The weight class could be set to `None` or `balanced`, which can make up for unbalanced datasets. Each run with a dataset ended up saving the best combination of each parameter to be saved based on the best accuracy offered.

2) XGBoost: `Scikit-learn` modules were used for splitting the dataset and grid search. XGBoost classifier was used for the training and joblib for saving the model and vectorizer. For consistency, 2000 word `TF-IDF` word limit was used too. To expand on Random Forest, not only the estimators (100 or 200) and the depth were tuned (4, 6, or 8), but also the learning rate (0.01 or 0.1) and the subsample (0.8 or 1.0).

3) XLNet: The XLNet implementation from the `transformers` library was used, in conjunction with `Scikit-learn` and `Torch` for data and model handling. All descriptions are tokenized using the pretrained XLNet tokenizer `xlnet-base-cased`. Parameters tuned for are learning rate (2e-5 or 3e-5) and patch size (16 or 32). Given the computational demand of the model and the relatively large scale of the dataset, the grid search is limited to few parameters.

4) BERT: Once more, `transformers`, for BERT the tokenizer, `Scikit-learn`, for data handling, and `Torch`, for deep learning. The data split is done differently here. The data is split in a training set (0.7), a test set (0.15), and a validation set (0.15). The parameters that the grid search is optimizing are learning rate (2e-5, 3e-5, or 5e-5) and batch size (8, 16, or 32).

D. Model training and evaluation

The computational demand of the models being trained in conjunction with the datasets is large, thus, XGBoost, XLNet,

and BERT, the most computationally heavy ones, were trained on the HPC cluster of the University of Twente. The Random Forest and CNN models were trained on a base Macbook Pro 14, equipped with the M1 Pro chipset.

To have a second layer of verification, a batch test is performed on the models with a 50 entry file generated by ChatGPT with the instruction to make the entries as random as possible, meaning they are not cybersercurity related. This provides insight into how the model would perform in real world cases where the input given can be unpredictable. Further, it can help give some insight into wether the model learned the wrong rules or simply memorized the entries. Henceforth, the entries generated by ChatGPT will be referred to as the synthetic entries.

For a more visual step by step of the Methodology, refer to Figure 5 in the Appendix. The data-related, blue steps apply universally, where as the green, model-related steps are repeated for each different model.

V. Results

Refer to Table III for clarification on the subsets.

A. Random Forest

Random forest was computationally light, relative to the other models. Figure 1 illustrates the findings. It provided acceptable levels of accuracy, over 80% across all three datasets. It handled the synthetic entries incredible well for both of the subsets, but abysmally for the full dataset.

The best result was seen in Subset2, with a 83% accuracy with the CVE dataset, and 98% with the synthetic one. The parameters that generated the best model with this subset were: 200 estimators, max depth of 20, and balanced class weight enabled.

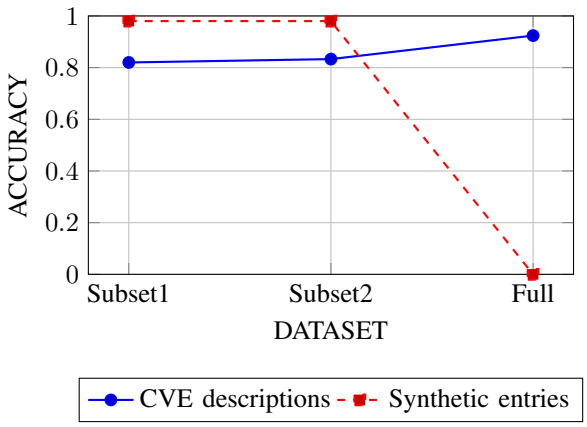

Fig. 1: Random Forest Model Performance

B. XGBoost

XGBoost was computationally more intense than Random Forest, but still light for the cluster. Overall, the results for this model are less than ideal, as seen in Figure 2. The accuracy however around 60% for both the subsets and only breaches 80% with the full set. However, the model has been incapable of handling the synthetic set, mislabeling it in its entirety. Within that context, the best model was the one generated by the full dataset, with 100 estimators, max depth of 4, a learning rate of 0.1, and a sub-sample of 0.8.

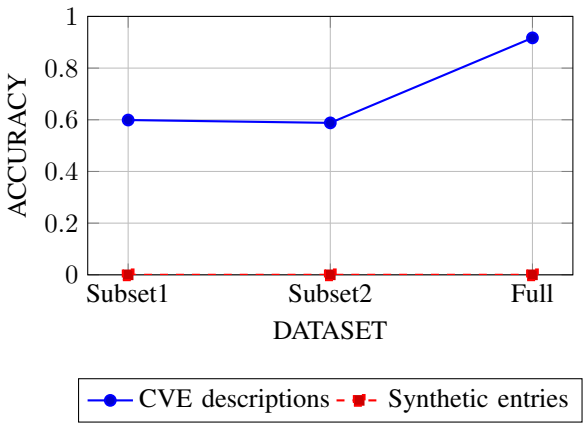

Fig. 2: XGBoost Model Performance

C. XLNet

Figure 3 shows the results of the XLNet models. All three models are rather accurate, ranging between 87% and 94%, but in spite the high accuracy, the performance with the synthetic dataset shows a decreasing trend as the training datasets increase. The best XLNet model is the one generated by Subset1, with an accuracy of 87% on the CVE set and 50% on the synthetic set. The best parameters for this model were learning rate 3e-5 and a batch size of 32.

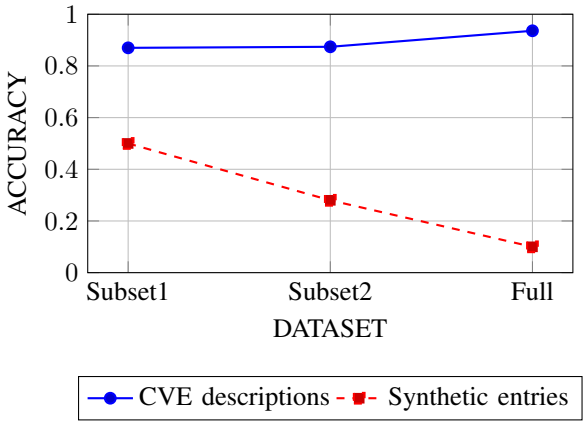

Fig. 3: XLNet Model Performance

D. BERT

The BERT models were the most computationally intense and took even the cluster some time to process. They provided high accuracy levels across all three models, as seen in Figure 3. The synthetic dataset proved to be a challenge for the model trained on the full dataset, making its 94% accuracy dubious. The model outputted by Subset2 is the best one, with an accuracy of 87% on CVE entries and 92% on synthetic

ones. The best parameters for Subset2 were a learning rate of 2e-5 and batch size of 16.

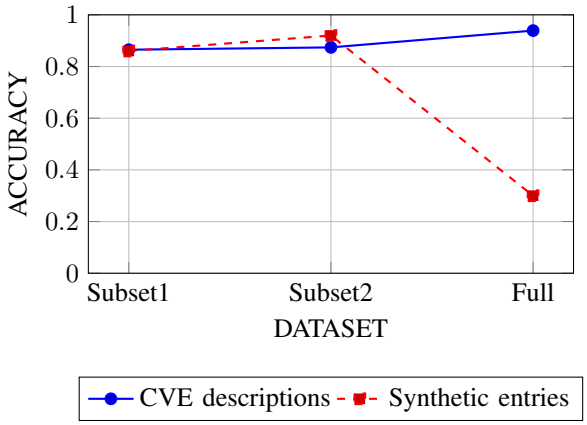

Fig. 4: BERT Model Performance

VI. Discussion

Even though it may not be immediately apparent, when making a decision on which models perform better in the context of this paper, the synthetic entry test is relevant to assess how the model reached its claimed accuracy percentage. To illustrate this point, refer back to the contrast between the accuracy of the trained models on the full dataset and its performance on the synthetic dataset. The full dataset provided the highest accuracy on CVE testing, whilst the offering the lowest in the synthetic tests compared to the models trained with the two subsets.

This peculiarity can be attributed to the fact that the dataset distribution is so skewed that even if a model always predicting `1` would be correct 89% of the time. Due to this, it was essential to test with reduced sets too. The original dataset was sufficiently large that even a reduced subset was more than adequate to train a model.

A. Model Evaluation: Inadequate models

XGBoost can be categorically stated as the worse performing model of the ones trained and tested, as the results suggest that the model mostly attempted make very biased rules towards the `1` classification, making it unusable in real world scenarios. Its extremely high accuracy in in the full model is misleading.

XLNet, given provided high accuracy and low, but not totally unredeemable synthetic testing accuracy. Even though this contrast proves it to be unreliable for the use case of binary CVE description classification, it could imply that the model itself could be suitable to this application given a different approach, with either data preprocessing or parameter tuning.

Lastly, even though CNN was originally drafted as a model to be explored, after testing one configuration with Subset2 which outputted an accuracy of 85.1%, it was discarded due to the vocabulary limitation. The way the used implementation operates is through making a vocabulary of the entire training, testing and validation sets before proceeding with the model training. This means that any future inference with this model has to be used with the vocabulary used to train the model, making it rather limiting in certain use cases.

B. Model Evaluation: Suitable models

Random Forest and BERT both performed remarkably well with the CVE dataset. In spite their vast differences in approach, implementation, and computation, they both offered similar reliability. BERT offered a marginally higher accuracy with CVE sets, whilst Random Forest was better at seething through the synthetic dataset.

Based on these tests, both can be considered reliable enough to assess whether a CVE description contains enough information to be labeled, whilst also avoiding falling for random strings - types of sentences it has not been trained on.

VII. Conclusion

CVE descriptions do not always contains sufficient information to be labeled with the appropriate CWE categories, but it is not currently possible to determine that automatically without expert input. From the findings in this paper, it is safe to state that CVE descriptions are suitable for binary classification through machine learning models. Not all models seem to be a good match, as seen with XLNet and XGBoost, but there are other that offer promising results, such as Random Forest and BERT, both in the range of 83% - 87% in terms of accuracy, whilst avoiding overfitting and learning appropriate rules that avoid being spoofed with by generic input.

In conclusion, natural language processing binary classification models could stand to aid in the assessment of CVE descriptions in order to facilitate further automation with the CVE datasets, such as automatic CWE labeling.

VIII. Future work

This work could have been enhanced by seeking input from a professional in the industry registering CVEs and their respective descriptions. They could offer insights into how the descriptions are composed and later assessed manually. Meaningful insights for the model design and data processing could have been drawn from it. Additionally, applying a training-testing-validation data splits to all explored models could make the results more comparable. Only some models were explored with this split in this research due to time constraints.

Furthermore, it a next step could be following up on some applications for being able to categorize CVE as labelable or not. One instance of this could be in cleaning the pre-existing but polluted CVE datasets. Another example could be offering feedback on why a certain CVE description is either adequate or insufficient.

References

[1] The MITRE Corporation, "About CVE Records," 2020, Accessed: 2025-05-04. [Online]. Available: https://cve.mitre.org/cve/identifiers/index.html

[2] E. Aghaei, W. Shadid, and E. Al-Shaer, "ThreatZoom: Hierarchical Neural Network for CVEs to CWEs Classification," in *Security and Privacy in Communication Networks*, N. Park, K. Sun, S. Foresti, K. Butler, and N. Saxena, Eds. Springer International Publishing, 2020, pp. 23–41.

[3] The MITRE Corporation, "History," Accessed: 2025-05-04. [Online]. Available: https://www.cve.org/About/History

[4] ——, "New to CWE," 2023, Accessed: 2025-05-04. [Online]. Available: https://cwe.mitre.org/about/new_to_cwe.html

[5] ——, "History," 2022, Accessed: 2025-05-04. [Online]. Available: https://cwe.mitre.org/about/history.html

[6] S. S. Das, E. Serra, M. Halappanavar, A. Pothen, and E. Al-Shaer, "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)*, 2021, pp. 1–12. [Online]. Available: https://ieeexplore.ieee.org/document/9564227

[7] J. Su and Y. Wu, "Refining CVE-to-CWE mapping with enhanced attention in BERT-based models," vol. 71, pp. 107–112, 2024. [Online]. Available: https://www.ewadirect.com/proceedings/ace/article/view/15315

[8] V. Yosifova, A. Tasheva, and R. Trifonov, "Predicting Vulnerability Type in Common Vulnerabilities and Exposures (CVE) Database with Machine Learning Classifiers," in *2021 12th National Conference with International Participation (ELECTRONICA)*, 2021, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9513723

[9] R. Oostveen. (2024) CWE-ASSIST: A framework for automating CWE classification. [Online]. Available: https://essay.utwente.nl/104547/

[10] K. Kota, M. A, and S. V. S, "CWE Prediction Using CVE Description - The Semantic Similarity Approach," vol. 235, pp. 1167–1178, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050924007877

[11] X. Sun, L. Li, L. Bo, X. Wu, Y. Wei, and B. Li, "Automatic software vulnerability classification by extracting vulnerability triggers," vol. 36, no. 2, p. e2508, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2508

[12] N. Arabadzhieva - Kalcheva and I. Kovachev, "Comparison of BERT and XLNet accuracy with classical methods and algorithms in text classification," in *2021 International Conference on Biomedical Innovations and Applications (BIA)*, vol. 1, Jun. 2022, pp. 74–76.

[13] M. N. Uddin, M. F. B. Hafiz, S. Hossain, and S. M. M. Islam, "Drug Sentiment Analysis using Machine Learning Classifiers," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 13, no. 1, 2022/55/31.

[14] J. Singh, G. Singh, and R. Singh, "Optimization of sentiment analysis using machine learning classifiers," *Human-centric Computing and Information Sciences*, vol. 7, no. 1, p. 32, Dec. 2017.

[15] Y. Bao, Z. Deng, Y. Wang, H. Kim, V. D. Armengol, F. Acevedo, N. Ouardaoui, C. Wang, G. Parmigiani, R. Barzilay, D. Braun, and K. S. Hughes, "Using Machine Learning and Natural Language Processing to Review and Classify the Medical Literature on Cancer Susceptibility Genes," no. 3, pp. 1–9. [Online]. Available: https://ascopubs.org/doi/10.1200/CCI.19.00042

Appendix

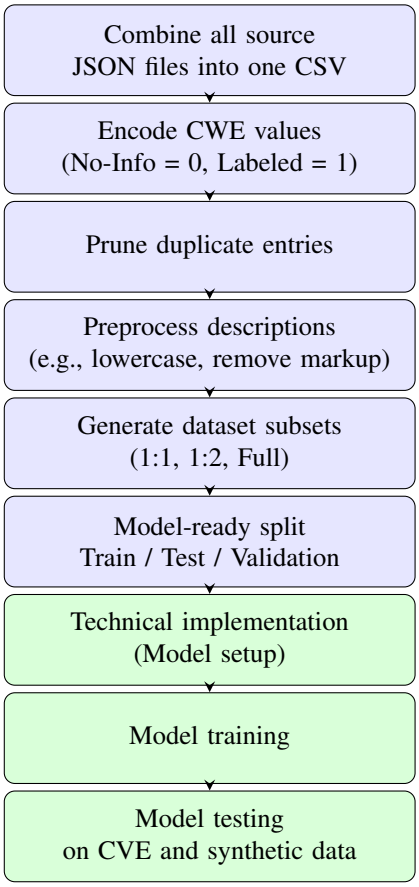

Fig. 5: Methodology Flow