Mapping Natural Language Security Text to CWEs using NLP techniques and SecureBERT vector embeddings

ALEX TOMA, University of Twente, The Netherlands

The fast growth of technology has opened ways of knowledge and sources of information to humanity but has also significantly increased the exposure to cyber threats. This increase also resulted in a rise of many vulnerabilities and weaknesses in software and systems, providing opportunities for malicious actors to exploit and compromise private data and disrupt key services. The study addresses the challenge of the lack of labeled datasets for CTI reports and security blogs.

The research aims to automate the process of extracting unstructured text security from natural language sources. The goal is to expand the usefulness of CWE classification beyond CVE, allowing more efficient classification of vulnerability-related security information. The methodology employs Natural Language Processing (NLP) techniques, specifically leveraging **SecureBERT vector embeddings** to represent the semantic content of security-related text from diverse sources such as **Cyber Threat Intelligence (CTI) reports and security blogs**.

The research aims to study the behaviour of various NLP techniques over the performance and interpretation of unsupervised learning using the KMeans clustering algorithm. It evaluates the performance of the clustering algorithm using the silhouette score. The study aims to sort out which extracted content represent a vulnerability-related text using the cosine distance metric.

Additional Key Words and Phrases: CVE(Common Vulnerabilities and Exposures), CWE(Common Weakness Enumeration), CTI(Cyber Threat Intelligence) reports, blog posts, NLP(Natural Language Processing), SecureBERT,vector embeddings

1 INTRODUCTION

Nowadays, the digitally connected world makes cybersecurity a top priority. Due to the complexity of the software and evolution of the cyberthreats, it is necessary to acquire strong mechanisms for recognising, comprehending, and mitigating vulnerabilities. In this process, 2 key frameworks play a meaningful role: Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration(CWE)[1].

CVE represents a database containing records about specific vulnerabilities discovered in both software and hardware. It serves as an indexing system of vulnerability, offering a unique identifier for each of them. CWE represents a framework that catalogs security vulnerabilities by storing information about its identification, mitigation and prevention[2].

The CVE-CWE mapping plays a very important role from multiple perspectives. The mapping offers valuable context for understanding the nature of the vulnerability. This mapping permits a profound understanding of the causes which led to the rise of the vulnerability, making it possible to find mitigation solutions. Moreover, the mapping facilitates a more efficient evaluation of the risks associated with a specific vulnerability[2].

While the CVE-CWE mapping empowers the communication between different actors from the cybersecurity field, there are also challenges and limitations that come along. The mapping leaves a gap in the ability to classify security information from its unstructured form.

1.1 Research Questions

RQ1: How does various NLP techniques influence the performance and interpretability of clustering algorithm?
RQ2: How can CVE-related information be effectively identified and extracted from blog posts and CTI reports?

This research aims to classify unstructured security text into vulnerability-related text using NLP techniques and SecureBERT vector embeddings, to identify vulnerabilities. Section 2 introduces a literature overview of the relevant studies related to this research study, Section 3 presents the methodological approach applied during this research. The findings of the research study are presented in Section 4 and discussed in Section 5. The research paper ends with a conclusion in Section 6 and future research recommendations in Section 7.

2 RELATED WORK

2.1 Literature review

Cyber Threat Intelligence (CTI) analysis is an important field of research in this domain, as it entails automatically extracting and categorizing information regarding cyber risks from diverse sources such as security blogs, threat reports, and social media.

Many works use standard NLP approaches such as TF-IDF and Word2Vec in conjunction with classical ML algorithms such as Support Vector Machines (SVMs), Naive Bayes, and Logistic Regression for problems such as threat categorization and topic modeling. Arazzi et al [2] present an extensive analysis of using NLP techniques in the CTI domain. The authors explore the way NLP techniques can be used to collect, process, analyse and share CTI information, underlining its meaningful role in identifying and mitigating cyber threats. Moreover, they elaborate the NLP techniques for extracting the CTI data from web sources and CTI reports.

Orbinato et.al [3] presents an experimental study about the automatic mapping of unstructured information about cyber threats (CTI) into attack techniques.The authors were using the MITRE ATT&CK knowledge base to create 2 datasets of CTI descriptions in natural language.

The authors of [7] focus on Docker containers' security. They identify that the misconfiguration of the containers during the implementation represent a major vulnerability source, letting attackers

to exploit the systems. Behzadan et al. (2018) developed a method to gather and classify Cyber Threat Indicators (CTIs) from Twitter streams. This system classified tweets as security-relevant or irrelevant using Convolutional Neural Networks (CNNs) on a binary and multi-class basis. Adewopo et al. (2020) studied open-source data for CTI and predicted the relevancy of postings from hacker forums and Twitter using Logistic Regression with TF-IDF. Due to the shortcomings of these conventional methods, deep learning techniques—in particular, transformer-based models like BERT and GPT—were investigated in an effort to enhance CTI analysis performance. FeedRef2022, a dataset for Named Entity Recognition (NER) in cybersecurity, was developed by Chan et al. in 2022. Using the FeedRef2022 dataset, the authors assessed how well modified BERT and other transformer models performed in extracting Indicators of Compromise (IoCs). BERT was investigated by Alves et al. (2022) for the classification of Tactics, Techniques, and Procedures (TTPs) in unstructured text. Using TRAM, an open-source platform that maps attack strategies from unstructured text, they were able to obtain greater accuracy than standard methods.

2.2 Research Gaps

Existing NLP and cybersecurity research has advanced significantly in tackling a number of issues. But there are still a number of research gaps that this study seeks to fill:

*2.2.1 **Automated CVE-to-CWE Mapping**.* The current techniques for mapping CVEs to CWEs are frequently rule-based or manual, which limits their accuracy and scalability. This study suggests automating this mapping via SecureBERT embeddings and cosine distance in a k-means clustering framework, which may increase accuracy and efficiency.

*2.2.2 **Using SecureBERT Embeddings for Clustering**.* Although SecureBERT has shown promise in text classification, its applicability to clustering applications is still mainly untapped. The goal of this study is to cluster vulnerability descriptions using SecureBERT embeddings in order to group related vulnerability-related text according to their semantic meaning as determined by a language model that has been specially trained on cybersecurity material.

*2.2.3 **Assessing SecureBERT Embeddings' Performance for Unsupervised Learning**.* The purpose of this study is to assess how well SecureBERT embeddings perform for unsupervised learning tasks, specifically k-means clustering for vulnerability detection. This will entail comparing the outcomes to alternative embedding techniques and evaluating the quality of clusters using measures such as the silhouette score. The study intends to automate CVE-to-CWE mapping and categorize related vulnerabilities according to their semantic similarities by utilizing SecureBERT embeddings and k-means clustering. This strategy may improve vulnerability management procedures' efficiency, which would ultimately lead to faster cybersecurity procedures.

3 METHODOLOGIES

This chapter outlines the methodological approach applied during this research. Typical steps applied are as follows:
1. Data extraction: collect a dataset of vulnerability text descriptions

Elements	Amount
web pages	10
Blog posts	184
CTI reports	495
CTI paragraphs	12511
blog posts paragraphs	3167
Total Paragraphs	15,678

Table 1. Amount of data extracted from website

from multiple sources including security blogs, and CTI reports. 2. Data preprocessing: use common NLP techniques such as tokenization, stopwords removal and stemming to clean and normalize the data.
3. Vector embedding: textual data will be transformed into vector embeddings using the SecureBERT framework.
4. Clustering: The Kmeans clustering algorithm will be applied to the preprocessed data and used to classify text segments.

3.1 Data Extraction

Data extraction is an essential process in data analysis, especially in the case of working with unstructured data such CTI reports text and blog posts. Table 1 presents a detailed overview over the dataset used in this research. The dataset consists in 495 CTI reports [6] and 184 blog posts from The Hacker News[1] website. The dataset contains only unlabeled data. In order to gather the data from the blog posts, a web scraper was implemented using the following python libraries: *Selenium* and *BeautifulSoup*. Fig.2 presents all the steps performed by the scraper in the extraction process:

(1) ***Create a connection to the main website***: the web scraper uses a headless Firefox webdriver. The website[2] is a HMTL page which needs to be parsed by the webdriver.
(2) ***Select div tag which stores all the blogs***: each page have in its HTML structure a <div> tag which delimits the main body from the rest.
(3) ***Select individual div tag of a single blog post***: the condensed version of a blog is delimited as inner box in main body.
(4) ***Look for the target link***: each of the inner box has a <href> tag which stores the link to the entire blog post
(5) ***Collect all links***: the program collect all the targeted links
(6) ***Create a new connection for each of the individual links***: for each of the collected links, the webdriver creates a new request and parse the HTML page (blog post)
(7) ***Select the article body***: every blog webpage contain a div tag which delimits the targeted data from the rest of the page
(8) ***Extract the paragraphs***: the program searches for the <p> tags inside the article body

---

[1]https://thehackernews.com/
[2]https://thehackernews.com/search/label/Vulnerability

3.2 Data preprocessing

Data preprocessing represents an essential step in preparing unstructured data for analysis. Cleaning and converting the raw data into the format that the ML algorithms can use are steps in this process.

3.2.1 Cleaning.

(1) Eliminating redundant characters: This entails eliminating punctuation, HTML tags, and special characters that do not add to the text's meaning.
(2) Lowercase conversion: guarantees that the same words are handled consistently regardless the capitalization

3.2.2 Tokenization.

(1) Word-by-word division: This entails dissecting the text into discrete words, or tokens, which serve as the fundamental analytical building blocks for NLP techniques. Sentence tokenization functions are offered by libraries such as NLTK.

3.2.3 Standardization:

(1) Stopwords removal: common words like "the" , "a" and "is" which are used a lot yet and have little semantic value are known as stopwords. Eliminating them can increase the effectiveness of analysis and lessen the dimensionality of the data. NLTK libraries are used by sources to eliminate stop words.
(2) Lemmatization/Stemming: Lemmatization transforms words into their basic form (lemma), whereas stemming reduces them to their root form. The consistency of analysis can be increased by using these strategies to group together various variants of the same term.

3.3 Vector Embeddings

Vector embeddings are numerical representations of text data that aim to capture the semantic and contextual meaning of words, phrases, and sentences. These embeddings in SecureBERT are designed specifically for cybersecurity-related information, giving dense vector representations that are extremely useful for applications such as grouping threat intelligence data.

In this study, the emphasis is on clustering the extracted CTI reports and blog posts in utilizing SecureBERT's extensive embedding capabilities rather than employing it as a comprehensive end-to-end text classification model. This method is consistent with the idea that text's semantic meaning can be represented by vector embeddings, which are then used to compute similarity.

The methodology for creating vector embeddings will be determined using SecureBERT embeddings as follows: every paragraph from the extracted data will be entered into SecureBERT. The vector embedding is represented by the CLS token which is a special token that is found in the final hidden layer of the Transformer encoder. This embedding is generated at a paragraph-level and has a 768-dimension.

3.4 Clustering

An unsupervised learning technique called K-means clustering is used to put related text input into groups.

(1) Determining the number of clusters: for this experiment, a number of 2 clusters was used.
(2) Initialising centroids: K-means starts by randomly selecting k data points as initial cluster centroids. These centroids are the average points inside each cluster.
(3) Based on their SecureBERT embeddings, vulnerability description pairs will be compared for similarity using the cosine distance. The cosine distance is a metric for measuring the dissimilarity of two vectors in a multidimensional space. It is developed from cosine similarity, which measures how similar two vectors are by taking the cosine of the angle between them. The cosine distance complements cosine similarity by focusing on the distance between vectors. The closer the cosine distance is to 0, the smaller the angle is between the vector embeddings and the more similar the vectors are, showing a higher degree of similarity between the texts represented.
(4) Assigning Data Points to Clusters: Using a distance metric, like cosine distance, which is frequently applied to text data, each data point is allocated to the cluster whose centroid is closest to it.

3.5 Metrics

3.5.1 Silhouette Score.
Silhouette score[3] is metric which is used for evaluating the performance of a clustering technique. The silhouette score metric is calculated using the formula:

$$silhouette\_score = (b - a)/max(a, b) \qquad (1)$$

where
a= the average distance between each point within a cluster, or the average intra-cluster distance
b= the average distance between all clusters, or the average inter-cluster distance.

The Silhouette score has a range of -1 to 1. A cluster is dense and well-separated from neighboring clusters if its score is 1. Overlapping clusters with samples extremely close to the surrounding clusters' decision border are represented by a value close to 0. The samples may have been assigned to the incorrect clusters if the score is negative [-1, 0].

3.5.2 Cosine Distance.
Cosine similarity quantifies the degree of similarity between two vectors in an inner product space. It determines whether two vectors are pointing in about the same direction and is calculated by taking the cosine of the angle between them. It is often used in text analysis to measure the similarity between documents. Cosine distance calculates the cosine of the angle between two vectors to determine how dissimilar they are.

$$cosine\_similarity = <X, Y> /(||X|| * ||Y||) \qquad (2)$$

$$cosine\_distance = 1 - cosine\_similarity \qquad (3)$$

where
<X, Y> = the dot product of vectors X and Y
||X|| = the magnitude of the vector X
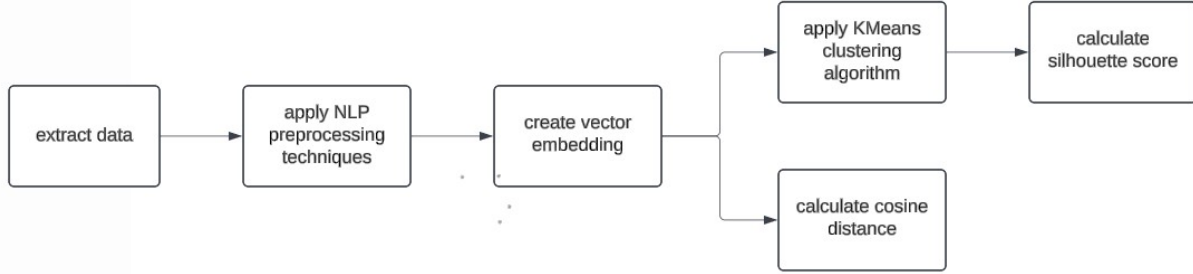
---

[3]sklearn.metrics.silhoutte_score

Fig. 1. Steps involved in the experiments

‖Y‖ = the magnitude of the vector Y

The resulting cosine similarity score indicates how similar two vectors are in terms of direction. A score of 1 implies that the vectors are aligned; 0 indicates that they are orthogonal (perpendicular), and -1 indicates that they are diametrically opposing.

4 RESULTS

4.1 Experiment 1

This experiment consists in grouping the extracted text into two categories:

- text which describes a vulnerability (vulnerability-related)
- text which does not describe a vulnerability (not vulnerability-related)

This research aims to see how NLP techniques impact the choice of the label of the K-Means clustering algorithm. For this, three forms of data are tested:

(1) raw data: data in the same form was extracted from the website
(2) semi-preprocessed data: data to which was applied punctuation and stopwords [4] removal
(3) preprocessed data: data to which was applied special characters removal, tokenization, lowercase, stopwords removal[3], stemming [5].

In this study, the 2nd form category was chosen due to the reason that punctuation marks and stopwords represent low-level information, meaning that the spotlight is on the specific security terminology. Moreover, SecureBERT is fine-tuned with an extensive dataset of specific security terms. By applying stemming, it may have a negative effect by changing the meaning of words, which may lead to inaccurate vector embeddings.

The purpose of the tests is to evaluate how NLP techniques influence the interpretation and the performance of the clustering algorithm.

---

[4]nltk.stopwords('english')
[5]nltk.PorterStemmer

Elements	Silhouette score
raw data	0.134
semi-preprocessed data	0.254
preprocessed	0.127

Table 2. Silhouette Score for the blog posts

Elements	Silhouette score
raw data	0.332
semi-preprocessed data	0.295
preprocessed	0.273

Table 3. Silhouette Score for the CTI reports

Table 2 and 3 make an overview of the scores of the conducted experiment.

4.2 Experiment 2

Since the silhouette score in experiment 1 is not close to its maximal value, this means that there still exists overlapping in between the vector embeddings which does not indicate a well-defined clustering.

Experiment 2 consists in picking 100 random CVE descriptions from the NVD dataset, embed them and calculate the centroid. Afterwards, calculation of the cosine distance in between the centroid and each of the vector embedding of the paragraphs is required. For this experiment, based on the results of the clustering algorithm, the vector embeddings from the categories which had the highest silhouette score were picked since the higher the silhouette score is, the more accurate the vector embeddings are which may lead to an accurate result. Tabel 4 and 5 make an overview of the results of the conducted experiment.

5 DISCUSSION

Experiments in the field of natural language processing (NLP) frequently seek to determine the best preprocessing techniques for various types of text.

Paragraph	Cosine Distance
"Particularly, the vulnerability exploits the ""combination of a carelessly-exposed MSMQ instance with misconfigured permissions that leverages BinaryFormatter can be reached from any host via HTTP to perform unauthenticated RCE,"" security researcher Sina Kheirkhah said."	0.0028
"""This combo allows for a good old unauthenticated RCE,"" the researcher added."	0.0053
Some of the other guidelines to reduce exposure are listed below -	0.0128

Table 4. Paragraphs with their corresponding cosine distance(blogs)

Paragraph	Cosine Distance
"To summarize CVE-2018-14847, since insufficient validation is performed on the device during reading and processing input, the file name string passed as an argument with special characters to an open system call will lead to directory traversal, hence allowing arbitrary file read on the device, and thus information disclosure."	0.0112
Another issue which allowed the abuse of MikroTik devices is the use of weak password encoding using the hash of username and the known salt '283i4jfkai3389' while storing user credentials [11]. Details accessed from the user.dat file can easily be decoded using the logic shown in Figure 14.	0.0090
"A piece of malware [A3] (a Windows PE binary) was reported in the wild exploiting the discussed vulnerability by disguising itself as a browser update for Windows [12]. Upon successful execution, this malware would connect to a MikroTik router over TCP port 8291 to compromise the device and inject the Coinhive cryptominer link when any HTTP request is made."	0.01175

Table 5. Paragraphs with their corresponding cosine distance(cti reports)

Experiment 1 investigates the impact of text preprocessing on blog posts and CTI reports. Based on the results of this experiment, the first research question can be answered. Since blogs seek to present information in a narrative format, they are renowned for being verbose and descriptive. The analysis shows that for this kind of text, Category 2, which entails semi-preprocessing, produces the best outcomes. Because blogs tend to be verbose, it is very important to maintain contextual meaning and nuances while preprocessing. Excessive simplification of a text might eliminate very important features, resulting in a loss of coherence and meaning. Applying Category 2 preprocessing allows the system to preserve enough information for precise analysis by finding a balance between raw and outrageously simplified text. In case of CTI, category 1 represented
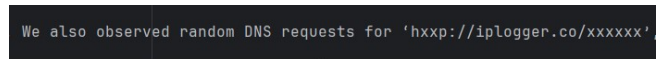

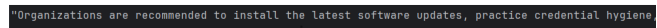Fig. 2. Sample of a sentence from a CTI report

Fig. 3. Sample of a sentence from a blog

by the raw text, produces the best results, meaning that none of NLP techniques applied in this experiment make an improvement. This may be happening because of the specific cybersecurity language usage. Fig.2 and 3 show how different language have both information sources.

Experiment 2 investigates how vulnerability-related paragraphs can be efficiently identified and extracted from the collected data. Based on the result from this experiment the second research question can be answered. Each of the CVE description consists in a series of elements which provide a informative overview of a vulnerability:

(1) **Affected system or software component**: it specifies which product, system or service is vulnerable; it may include a vendor name or a particular version
(2) **Nature of vulnerability**: it describes the root cause or flaw of the issue
(3) **Potential attack vendor**: it indicates how the vulnerability can be overburdened
(4) **Impact of the vulnerability**: it underlines the consequences or outcomes in case of the vulnerability is exploited

In order to accurately specify which paragraph is vulnerability-related, a threshold is needed. Based on the example below, a threshold can be defined.

The paragraph "*Particularly, the vulnerability exploits the "combination of a carelessly-exposed MSMQ instance with misconfigured permissions that leverages BinaryFormatter can be reached from any host via HTTP to perform unauthenticated RCE," security researcher Sina Kheirkhah said.*" has a cosine distance of 0.0028. This paragraph can be considered because it covers all the elements from the above.

(1) **Affected system and software**: "carelessly-exposed MSMQ instance"
(2) **Nature of vulnerability**: misconfigured permissions and BInaryFormatter usage
(3) **Potential attack vendor**: "can be reached from any host via HTTP"
(4) **Impact of the vulnerability**: RCE(Remote code execution)

The paragraph "*AI systems often involve complex algorithms, vast amounts of data, and can have far-reaching impacts on users and businesses.*" has a cosine distance of 0.0093 and it may be considered a bottom-line. In this experiment, for a more accurate identification,

it is needed for a paragraph that it contains at least 2 of the elements from above, which is not applicable in this case.

In case of the CTI, because of its very unstructured style, it is harder to define a threshold.

6 CONCLUSION

The results show that using language models like SecureBERT that have been specially pre-trained on cybersecurity text can improve the effectiveness of unsupervised learning methods in the field of cybersecurity. This strategy is in line with the expanding corpus of research showing how well some of the NLP approaches work with blog posts paragraphs. SecureBERT embeddings can be clustered using k-means clustering, which ease the requirement for large amounts of labeled data by grouping related vulnerability descriptions. This method provides a scalable and effective way to automate vulnerability classification.

The study's conclusions can be used practically in a number of cybersecurity fields, such as:

1)Vulnerability Management. By integrating the automatic CVE-to-CWE mapping system with vulnerability management tools and procedures, vulnerability evaluation and prioritization may be done more accurately and efficiently.

2)Threat Intelligence Analysis. By grouping related security risks together, security analysts may better analyze threat intelligence data, spot new threats, monitor attack campaigns, and create efficient defenses.

7 LIMITATIONS AND FUTURE WORK

The method presented in the research does have some limitations. The web scraper which is used for extracting data from the website assumes that all the blogs have a similar structure. This means that all the text inside the blog is stored in the <p> tag. In some cases, some blogs have different tags such as <li> tags which are avoided in this study.

Future work on this subject's performance offers several possible areas of improvement, such as:

1)**Managing Specificity in CVE Descriptions.** The model's capacity to generalize is hampered by the frequent use of extremely detailed terminology and technical jargon in CVE descriptions. By concentrating on crucial semantic components, methods like key term extraction and embedding may improve mapping accuracy.

2)**Investigating Different Approaches.** Alternative strategies that could boost performance include fine-tuning LLMs like GPT for direct CWE prediction, utilizing key phrase extraction, and fine-tuning instructor models for improved embedding creation.

3)**Absence of complete and labeled Datasets**. The model's capacity for training and assessment is restricted by the lack of comprehensive labeled datasets for security blogs and CTI reports. Future studies in this field would greatly benefit from the creation and open sharing of such datasets. By addressing these drawbacks and investigating different approaches, the suggested strategy can be made even more effective, opening the door for automated vulnerability management solutions that are more reliable and accurate.

REFERENCES

[1]K. Kota, M. A, and S. V. S, "CWE Prediction Using CVE Description - The Semantic Similarity Approach," Procedia Computer Science, vol. 235, pp. 1167–1178, May 2024, doi: https://doi.org/10.1016/j.procs.2024.04.111.

[2]M. Arazzi et al., "NLP-Based Techniques for Cyber Threat Intelligence," arXiv.org, Nov. 15, 2023. https://arxiv.org/abs/2311.08807 (accessed Dec. 18, 2023).

[3]V. Orbinato, M. Barbaraci, R. Natella, and D. Cotroneo, "Automatic Mapping of Unstructured Cyber Threat Intelligence: An Experimental Study," arXiv (Cornell University), Jan. 2022, doi: https://doi.org/10.48550/arxiv.2208.12144.

[4]Daegeon Kim, Huy Kang Kim, September 24, 2019, "Cyber Threat Intelligence (CTI) dataset generated from public security reports and malware repositories", IEEE Dataport, doi: https://dx.doi.org/10.21227/dpat-qd69

[5]"Vulnerability | News & Insights," The Hacker News.https://thehackernews.com/search/label/Vulnerability

[6]MaxWijnbergen, "GitHub - MaxWijnbergen/ Thesis_BIT_-MOD12," GitHub, 2024. https://github.com/ MaxWijnbergen/Thesis_-BIT_MOD12 (accessed Nov. 21, 2024).

[7]V. B. Mahajan and S. B. Mane, "Detection, Analysis and Countermeasures for Container based Misconfiguration using Docker and Kubernetes," IEEE Xplore, Jun. 01, 2022. https://ieeexplore.ieee.org/abstract/document/9885293

[8]Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer,L.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018) https://doi.org/10.48550/arXiv.1802.05365

[9]Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)

[10]Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pretraining of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[11] Beltagy, I., Lo, K., Cohan, A.: Scibert: A pretrained language model for scientific text. arXiv preprint arXiv:1903.10676 (2019)

A APPENDIX

A.1 Github

To see the code enter the following github:
https://github.com/althomx1/m12project