MSDS 686 Deep Learning

Final Project Write Up

Student: Eugene L. Moore

Date: 10/14/2023

I. Introduction

The National Vulnerability Database (NVD) contains a data set of cyber security vulnerabilities with metadata associated to security research work. The primary problem with this data set is the lack of standards for CVE text description. This makes using the data set as a source of solid reference information impractical for identifying vulnerabilities in software. The CWE data set on the other hand is a well-known and well-articulated set of weaknesses documented in a public data set that pertains to weaknesses found in software and hardware. Security researchers commonly use both, and other, data sets when performing exploitation research. Many data scientists have attempted to map CVEs and other open-source data sources to exploit data with varying success in terms of industry use. However, cybersecurity researchers are aware of these data sets features and limitations and how to use them intuitively in security research work. This project attempts to enhance the usability and efficiency of the CVE vulnerability data set by building upon a data set published in 2021 on Kaggle which maps CVE data to CWE data. In this experiment I will attempt to train an NLP model to accurately learn if CVE metadata is relevant to CWE. If this experiment succeeds the model could then be used to make a recommendation tag on new CVE metadata field to signify if it's relevant to a CWE based on this supervised learning experiment. Essentially, this model experiment attempts to capture the intuitions of a security researcher.

II. Overview of Experiment

This is a large project requiring data engineering, analytics, and several techniques of Machine Learning. This project incorporates the tools and techniques I've learned over the past year of the graduate level Data Science studies at Regis University. This project builds upon fundamentals of data engineering by using techniques and tools learned from the Data Engineering course as well as advanced techniques and tools learned from the Text Analytics course. The machine learning portion of this project builds on the tools and techniques learned in the Deep Learning course. The foundation of this project is the use of a neural net to solve an NLP classification problem. The highlight of this project is the demonstration of transfer learning. The inspiration for the transfer learning focus was built upon an article I researched during one of the discussion prompts in the course. I rely heavily on this article [2] for most of the logic to implement the BERT model. The remainder of the project and its problem use case is my contribution to the field.

I use a transformer architectural neural net called BERT. BERT stands for Bidirectional Encoder Representations from Transformers. BERT is a state-of-the-art ML model for NLP tasks. It was developed by Google AI labs in 2018. [3] I chose BERT because of its ability to solve polysemy[1] problems i.e., it can differentiate words that have multiple meanings based on the surrounding text.[3] My hypothesis is that I it is best suited to perform CVE metadata mapping to potential CWEs. This task is also difficult for humans that have been trained in the cyber security field for several years.

This classification problem is a categorical classification across multiple labels which is one of the most challenging NLP tasks when compared to common sentiment analysis although BERT is good at that too. The project is implemented in two Jupyter notebooks. The first Jupyter notebook contains the logic needed to prepare an open-source dataset [1] for machine learning in a neural net. The second notebook is used to process the prepared dataset for use in BERT. The next few sections provide greater detail on each area of the project.

[1] Polysemy (/pəˈlɪsɪmi/ or /ˈpɒlɪˌsiːmi/;[1][2] from Ancient Greek πολύ- (polý-) 'many', and σῆμα (sêma) 'sign') is the capacity for a sign (e.g. a symbol, a morpheme, a word, or a phrase) to have multiple related meanings.

a. *Notebook(s) used and why* - This project requires the use of two separate notebooks. The reason for this is due to using my own hardware, with its limitations, to complete the course final project. This required the use of two separate python environments. The primary driver for this was the GPU requirement for the Deep Learning (DL) portion of the project. Due to TensorFlow and Windows compatibility issues I was forced to use an older version of the CUDA library which was not compatible with the latest libraries needed for the Spacy library.

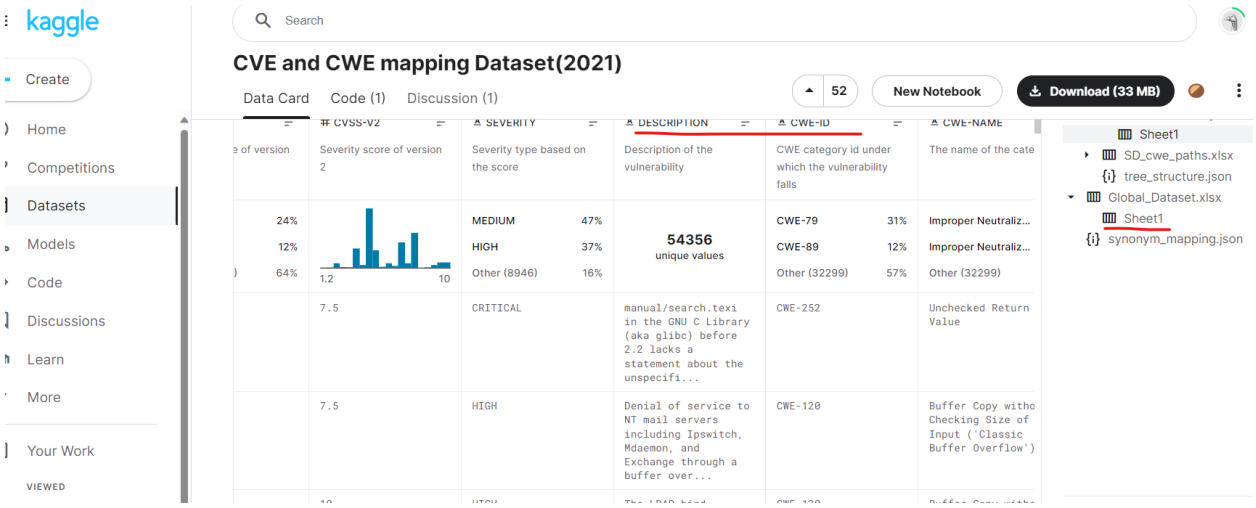
b. *The first notebook* - is for data prep. I chose to use the Spacey library for data cleansing. I could have used or wrote my own cleansing functions, however, I opted for a more robust framework due to its features and capabilities. I often do this as a principle of software engineering. It is much better to use reputable frameworks rather than to create your own. Although sometimes it is necessary to create your own; I avoid this if possible. I only choose to create my own functionalities for reasons that are driven by performance requirements or if the functionality simply does not exist in a tried and tested framework.

c. *The second notebook* - is for the deep learning portion of the final project in this notebook. I'm using the TensorFlow 2.13 framework which is compatible with my Windows hardware. The reason for using an a slightly older framework is due to the support for CUDA which allows me to use my GPU.

d. *Frame works and Libs* -For this project I'm using Anaconda 23.74 with two separate virtual Python environments. I've enclosed the requirements text for each of the environments. The first environment is called NLP latest which is the environment used for the data preparation notebook. The second requirements text is the tensor flow TF 2.13 environment which contains an earlier version of TensorFlow which is compatible with my windows machine. In addition to the requirements text files created for this project, I've also created an environment configuration file called NLP latest which is a YAML file. This can be used to quickly create

a virtual environment for the latest data preparation tools needed to perform the experiment, as detailed in this project. As for the later version of TensorFlow 2.13 I chose not to create a YAML because it will not be needed as I will transition most of my work to a workstation that is Linux based and more compatible with deep learning frameworks.

e. *The dataset* - I chose for this project is from Kaggle and it is the CVE and CWE mapping data set of 2021. This data set is relatively recent and up to date. The link to the dataset is listed in references. As shown in the figure below, I've underlined two features that I chose to use for my project. These two features represent the label and target. The description text provides metadata relating to the type of vulnerability that was recorded in the CVE vulnerability dataset. The CWE ID relates to the weakness enumeration from The CW E dataset. The purpose of this dataset and experiment is to train a model to be able to identify CWE IDs based on the metadata of a CVE vulnerability description text. It is important to know that CVE vulnerability data is an open-source database that security researchers contribute to when they discover a cyber security vulnerability. It is often criticized for its lack of standards based due to the acceptance of sketchy description data. This makes it very hard for a cyber security analyst to effectively utilize the metadata when researching vulnerabilities discovered by scanners that report on CWE IDs. By training a model to identify possible CWE IDs based on the CVE description metadata can provide away to identify potential CWEs related to the description of a vulnerability in the CVE database. The model created in this project could greatly increase the efficiency of mapping these two datasets since the rate of increase for CVEs is exponential when compared to the relatively static CWEs. The relationship between CVE and CWE ID is almost 1 to many.



II. Data Preparation

In the first section of the project Data Prep Jupiter notebook I import the pandas library to read in the CSV file for the dataset chosen. The first section provides basic functions to perform some Exploratory Data Analysis (EDA) and as shown at input 15 the data set is large in terms of features. I only need two features for this experiment which are CWE ID and Description. I continue to explore the dataset to see how many value counts there are as shown at input 16. After this exploration I choose to prep the data for a binary classification problem. This is because each CWE ID represents a CWE type. Note: not all CVE descriptions map to a CWE type. By doing this I can now identify whether the CVE data is in fact targeting a CWE. Once more I have dropped all of their targets at input 39. I'm now left with a new data frame with two features: a description and a one or zero for a CWE type. So, either the description in the CWE is talking about a CWE or it is not. In the following next section, I clean the text. I now import the Spacey library and I use the large web data set for NLP. From here I write a quick clean text function to perform a Lambda operation on the description text. Once this is complete, I combine the two features into my project data frame and save it to a CSV. In the final section I perform a training split to view the data and make sure that it will support my deep learning experiment in the next Jupiter notebook.

III. NLP Classification

Now that my data engineering has been completed I import the needed libraries to import the prepped CSV file and perform a quick observation to ensure the data is intact. Next, I create the necessary variables for the description text and the labels and observe the shapes. I also print out the description to observe the description text. At input[7] I perform my train test split and at input[8] I import the BERT model to begin tokenization and configuration for my experiment. I use BERT to encode the words in the description text and I print them to observe the word embeddings at input[10]. Next I need to find padding. Padding is used to ensure standard input sizes for the description text. So, for example, some description text can vary in length, so I need to find the Max length to create the correct padding. This is performed in the section Find Padding values. Once this is complete, I need to find my mask input. This is done with the description text and the Max length for description text. I do this in the function at input[13]. This will also prepare the necessary encoded dictionaries to create my training test split shown at input[14]. I can now print out the description text to identify the necessary sections to feed to the BERT model just like in the article. You can see the separation between the sentences by observing the encoded values for the beginning of the sentences and the ending of the sentences. The 101 encoding at the beginning of the dictionary signals an input and the 102 encoding signals the end. In between the array you can see the word embeddings. Also notice the clean text descriptions and some of their ambiguities. This is why it is a hard task to map this type of sparse description to a CWE type. Next, I import the BERT sequence classification model. Following the article instructions, I set up a log directory and a save path for my model. For this experiment I choose to train all weights with this input data. I print the summary of my model at line 50 and you can see that my total parameters for training are 109,483,778 at input[51]. I now attempt to fit the model. In this experiment I was unable to use a batch size of 32 due to limitations to my hardware. With some quick research I was able to troubleshoot the problem and as you can see at input[51] the largest batch size I could use is 2 however over 4 epochs I was able to get reasonably good

results in terms of accuracy and loss. Finally, I create a function to plot my history metrics and at input[54] I plot the history results. As you can see on the training and validation accuracy plot I quickly converge and it seems to be will fitted when compared to the training loss.

IV. Conclusion

I will discuss experiment results and possible follow-on work. I consider this project experiment a success in that I was able to implement transfer learning with the BERT NLP model. Although I experienced some difficulties with my personal computer hardware, I learned from the experience which I consider to be more valuable than theoretical knowledge in the field of AI. The next possible test project would be to determine how effective my model is if tested on a data set from the NVD database. If it could successfully categorize or tag description text with a CWE recommendation this would allow for faster classification of new CVEs that target software vulnerabilities from 2021 forward.

Another project I would like to pursue related to this one would build upon this model experience to create a model that could then identify CWE ID categories. This would be a much more ambitious project. This is because there are greater than a dozen CWE IDs. However, I am very happy with the result of this experiment since the CVE data set is quite large and being able to at least tag CVEs as a possible candidate for CWE relevance is a positive step in terms of efficiency. This model could be used to augment deficient metadata the NVD is known for.

V. Reference

1 Kirushikesh, D. B. (2021). *CVE and CWE mapping Dataset (2021)* [Data set].

2 Singh, Y. V. (2021, August 18). Classification using Pre-trained Bert Model (Transfer Learning). Medium. https://medium.com/@yashvardhanvs/classification-

using-pre-trained-bert-model-transfer-learning-2d50f404ed4c

3 *BERT 101 - State Of The Art NLP Model Explained*. (n.d.). Huggingface.Co. Retrieved October 14, 2023, from https://huggingface.co/blog/bert-101

4 Wikipedia contributors. (2023, October 4). *Polysemy*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Polysemy&oldid=1178572777

5 *NVD - Home*. (n.d.). Nist.Gov. Retrieved October 14, 2023, from https://nvd.nist.gov/

6 *CWE - CWE List Version 4.12. (n.d.). Mitre.Org. Retrieved October 14, 2023, from https://cwe.mitre.org/data/index.html*