

Classification of PII and Non PII files using Machine learning (NER) for Data loss Prevention

MSc Research Project
MSc Cyber Security

Shivraj Prithviraj Mohite
Student ID: x21171327

School of Computing
National College of Ireland

Supervisor: Prof. Apurva Vangujar

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Shivraj Prithviraj Mohite

Student ID: x21171327

Programme: MSc Cyber Security **Year:** 2022-2023

Module: Research Project

Lecturer: Prof. Apurva Vangujar

Submission Due Date: 14th August 2023

Project Title: Classification of PII and Non PII files using Machine learning (NER) for DLP

Word Count: 7117

Page Count: 22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Shivraj Prithviraj Mohite

Date: 14th August 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Classification of PII and Non PII files using Machine learning (NER) for DLP

Shivraj Prithviraj Mohite
X21171327

Abstract

Data breaches have now become quite common due to the rise of production and procession of sensitive data by organizations. This Sensitive data is called PII or Personally identifiable information which when leaked could be point to a particular individual and put the privacy of that user at risk. So, the detect such PII information and prevent it from getting leaked publicly organization's use Data loss prevention system to detect these PII information and prevent them from exfiltration. These PII information could be their employees data or their clients information in order to protect these kinds of data they purchase pricy DLP tools which are able to classify and detect the PII files using keywords based classification, regular expression etc but the traditional legacy DLP systems are not that accurate so to improve the quality of detection I have introduced a new Machine learning based approach which uses Named Entity Recognition which is NLP method which extracts information from text. In our thesis we have used a pre trained BERT model which have finetuned for NER to successfully classify the PII and Non PII files with accuracy of more than 92%. This approach can be more effective in detecting PII files than the traditional legacy DLP approaches.

Keywords: NER: Named entity recognition, NLP: Natural language processing, BERT model, machine learning, classifier, python, transformers, spaCy, PII, Non PII.

1 Introduction

1.1. Background

Data privacy has become a major concern for individuals as well as the organizations in today's data-oriented world. Personally Identifiable Information (PII), such as credit card numbers, driver's license numbers and medical records, are often targeted by cybercriminals but there are more components which are there which are not focused by the organization to protect like name, data of birth, location if information like these are tagged together with the Sensitive PII information then cyber criminals could be able to personally identify a specific person on globe and could perform their malicious intent like impersonation, bank fraud etc. Overall, there are two types of PII information Sensitive PII and Non-Sensitive PII. Non-Sensitive PII information is which is publicly available in records like name, date of birth, race etc which can be transmitted over the internet without any encryption as it won't harm the individual in any mean on the other hand there is Sensitive PII data like social security numbers, credit card numbers or any government id which cannot be sent unencrypted over the internet as it exposure could be very harmful for the individuals or the organization as

cyber criminals could use that information for any malicious intent. The legacy DLP systems are rule based and they generate lot of false positive detection and sometimes they are not able to classify the PII data overall which may lead to data exfiltration and data breach and these kind of breaches can cause significant damage to the companies reputation which may lead to huge financial loss and difficulties in getting new business hence Data loss prevention is treated as a High priority tool in every companies Cyber security architecture. As we know the European Data privacy law known as GDPR is so strict about processing of personal information and its exposure to public. Any data controller who failed to abide the GDPR law is penalized heavily and to be compliant with GDPR its important for the companies to focus upon preventing Data breaches. (Bernstein, 2023)

1.2. Motivation and Objective

The motivation to do research in this area was the lack of efficiency and accuracy of legacy DLP tools which only focus on Sensitive PII information leaving back the Non-Sensitive PII to get exposed. It is important to protect both Sensitive and Non-Sensitive PII data because if they are exposed then the attackers would be able to pinpoint a specific person and then perform their malicious activity as they would know whom they are targeting. Also, Critical Start survey revealed that 68% of SOC teams reported that 50-75% of the alerts they investigate are false positives (Clarke, 2023). In a real-world scenario, I know we can't eliminate false positives totally but there should be a defined percentage of how much false positives is acceptable. In my previous organization I have worked upon both DLP and SIEM tool as a SOC analyst and I have experienced that 75% of the alerts generated are false positives and almost 24% are benign positive (which are expected to happen, or which are approved by business) and there is only 1% of True Positives detected. All of these factors depend on the quality of configuration of the Data Loss preventions systems and the rules defined in them. During my time with the company, I realised that there were no special rules or policies to detect the Non-Sensitive PII information, as they were focused on Blocking Sensitive PII. We all know there is no need to block Non-Sensitive PII information until it has any sensitive data but at least there should be a system to keep track on any kind of PII information which is being transferred outside the organization. Hence, I have proposed a new machine learning based approach which would be able to classify between PII (Sensitive and Non-Sensitive) and Non PII files with more accuracy.

The primary Objective of the thesis was to develop an efficient, reliable and accurate machine learning based model which would be able to Classify between PII and Non PII files and later import that trained model into a Windows application to show real world working example of Classification between the PII and non PII files , addition to that I have designed the application to classify the files in 3 categories : Sensitive PII , Non Sensitive PII and Non PII. The objectives of the thesis are outlined below:

- 1) Investigation of Nature of PII and Non PII data: Understanding the nature, elements that differentiate the PII and Non PII data and challenges in classifying such type of data.
- 2) Developing Machine learning Model for Classification: In our thesis we have used the pretrained BERT model and finetuned it with a dataset of 18lakh entries which

contained both PII and Non PII data for Named Entity Recognition and text classification.

- 3) Evaluation of Models Performance: Comparing the developed models performance using various metrics to determine the effectiveness and accuracy of the model.
- 4) Importing the finetuned model in Windows Python Application: Importing the final finetuned BERT model and its tokenizer into the python application which will classify the content of the text file and determine whether they are PII or not.
- 5) Exploring the Applicability of Model in DLP: Analysing how the final model or application can be used in Data loss prevention and understanding its impacts on data security.

1.3. Scope

The Scope of the thesis includes the below points:

1. **Dataset:** The main part of creating or finetuning a machine learning based model is getting a dataset with proper labels and in a format which would be understandable to the model. In our implementation we have taken a dataset of over 18lakh entries which has both PII and Non PII information available along with their labels. This dataset was created by me using AI tools as its not possible to get dataset with PII information.
2. **Technology Used:** The thesis mainly relies on the Pretrained BERT model which is Bidirectional Encoder Representations from Transformers which is a state-of-the-art model for variety of NLP tasks. The BERT is pretrained on more than 2500million words.
3. **Application:** The primary application area is DLP and how the model can be used for more effective detection in DLP Tools for that I have created a Windows application to show of actual detection and classification between PII and Non PII files.

1.4. Thesis Outline:

- a. Section 1: Introduction: Introduces to the research problem, motivation behind the research problem, its objectives and scope.
- b. Section 2: Literature Review: Analyses the existing literature related to our subject like PII classification, Named entity recognition, BERT model, DLP etc.
- c. Section 3: Methodology: Describes the research design, data collection, model selection, training, and evaluation metrics.
- d. Section 4: Implementation:
- e. Section 5: Evaluation and Test cases
- f. Section 6: Conclusion and Future Scope
- g. References:

2 Literature Review

In this section we take a review of previous work which is relevant to our thesis topic.

Youngja park's (Park, 2010) approach to detect confidential documents through text mining consist of 2 major steps. First would be the machine learning based algorithm used to

identify documents which are labelled as confidential. This was achieved by them by using pattern-based matching to recognize canonical cue phrases. The label classification process determines whether the labels classify the documents confidentiality or can be used for any other purpose. Total 13 features, including label type, capitalization, negation within the documents and the sentences were used to identify or classify each of unique labels. The 2nd step involved using of fingerprinting technology for textual data to detect duplicate or closely duplicate files from this approach I got the idea of labelling a certain type of data in dataset with unique labels which could be used to train our machine learning based BERT model.

In the paper published by (Guha et al. 2021) regarding Self learning intelligent information leak protection system that uses a image mining and extraction technique to categorize or label documents or files as sensitive or non-sensitive based on the presence of NPI (non-personal public information) and PII. The protection system is trained and tested with production document images tagged as sensitive or non-sensitive which are classified on basis of presence of confidential data in it. The study also found that an artificial neural network-based approach using multilayer perceptron model outperformed many algorithms designed to detect the sensitive files. This paper motivated me to further divide the classification of PII files into Sensitive PII and Non-Sensitive PII and Non PII which added more security value to my thesis.

In a study conducted by Liu et al. (2020) on data leakage prevention technique using AI has presented us with a more sensitive data loss prevention technology which uses AI algorithm which can self-learn and detect. The system was designed to incorporate a prebuilt security policy that could detect confidential data and could report it to security manager for overview. This research included 3 key components which were Prebuilt security policy which would detect sensitive data and report the results for assessment to the manager for generating scores of accurate and inaccurate detections. The next one was Measurement model which establishes uniform measures of document important to upgrade the existing security policies and the last one was Batch Gradient descent Algorithm which was used to obtain optimal solutions for detections model and to create and specify new cyber security strategies.

While getting information about how we can classify or label each word or attribute in our dataset I saw one paper published by (Patil et al.,2020) where they have introduced creation of machine learning based model which will be able to classify between Personal, sensitive personal and non-personal documents gave me the idea about my research topic. In this particular research the author has used html tags for labelling the data and has use a model which consist of five phases which includes data curation, model selection, model evaluation, confidence metric and N-lines also they have used OCR techniques for data curation where they collect diverse file formats and convert them into text using OCR techniques but these techniques are not effective and may have many flaws which could create multiple issue in data curation resulting in degrading the quality. The use of <html tags> to manually label the file helped me with the idea of manually labelling the file in my own dataset which custom tags and later use that dataset to finetune the BERT model.

As the importance of Data privacy is increasing day by day along with the need for being compliant with EU data privacy law GDPR, the paper by (Silva et al., 2020) demonstrates an innovative approach of identifying, monitoring, and checking PII. In this paper the developers have introduced the use of Named entity recognition to detect data privacy violations. The research includes evaluation of 3 prominent NLP tools including NLTK, Stanford NER showing their applicability in the context of Data privacy. The paper is focused on the use of machine learning and NLP in context of PII for data privacy. As the paper focuses on use of NLP ML for pII detection it adds a critical dimension to evolving field of DLP providing more insights into both technical methods and ethical considerations involved in handling the PII data. In the context of currently developing text mining approach, self-learning models this paper is focused upon Named entity recognition and explains the potential of NER in order to enhance security and protect exposure of PII files.

In a study by Alexander Friebely, he demonstrates the use of Named entity recognition (NER) using Microsoft Presidio for the detection, classification and protection of PII files across different industries. The paper helps us understand the legal obligations on PII protections and helps us to evaluate the efficacy of Microsoft Presidio which has been used as Safeguarding method to protect the PII files. The research also shows lack of federal law in terms of data privacy in US and shows how GDPR is more effective. Where it also demonstrates that Named Entity recognition can be used to successfully protect or safeguard PII information. His work gives us better understanding of the use of NER in protecting PII data and also helps us understand how we can implement more AI driven technologies to Safeguard our PII information. (Friebely, 2022)

In today's world where personal information extraction has become a critical aspect of various applications, the research paper by (Hathurusinghe et al., 2021) al focuses on privacy preserving approach which utilizes both automatic annotation and federated learning. The authors have introduced WikiPII which would be an automated dataset composed of Wikipedia pages annotated for PII extraction. But these types of automated labelling system would create a tons of label noise due to its automatic nature, but the paper emphasizes on cost effectiveness of the solution which create large dataset with automation. The research uses the Same BERT model for Named entity recognition which is trained upon WikiPII in their concern which shows that large training dataset can decrease the cost of manual PII information extraction even when there is lots of false positive label noise. The paper also introduces us to federated learning which is a decentralized training system that enables model training over different platforms. The text mining approach which has been introduced for creation of auto labelled dataset will reduce human efforts of manual labelling of data and later the model can be used in the field of data privacy.

In a paper published by (Pearson et al., 2021) on the user Named entity recognition in Unstructured medical text document they have addressed the challenge of de-identifying independent medical examinations (IME) reports which may contain sensitive information, The manual process of deidentifying would be very ineffective and time consuming to over

come that they introduced NER based deidentification. The authors employed NER and open-source toolkits like OpenNLP and spaCy to automate this process, in their study they have found that spaCy model achieved the highest recall and f-measures demonstrating its feasibility of using NER For automating the deidentification. (Pearson et al., 2021)

Research Niche:

The summary of all the literatures review done above is presented in the below table with 4 key points: Paper name, Key focus, Limitation and Expected Contributions.

Table 1: Research Niche

Paper Name	Key Focus	Limitations	Expected Contribution
A Text Mining Approach to Confidential Document Detection for Data Loss Prevention	Develop a model that can detect general PII data without being specific to any company	Limited availability of annotated data, difficulty in achieving high accuracy due to the complexity of data	A comprehensive and generalized PII detection model that can be used by any organization to protect personal information
Data Classification using Deep learning	Develop a machine learning model that can accurately classify documents into personal, sensitive personal, and non-personal categories (Patil et al., 2020)	Difficulty in achieving high accuracy due to the complexity of data and variability in file formats	A comprehensive and accurate machine learning model that can classify various file formats and protect personal information from leakage
Self-learning Intelligent information leak system	Develop a self-learning and adaptable system that can identify and categorize sensitive and non-sensitive documents without explicit rule configuration (Guha et al., 2021)	Limited availability of annotated data, difficulty in achieving high accuracy due to the complexity of data	A proactive and real-time information leak protection system that can identify and categorize sensitive information without relying on explicit rules
Leakage Prevention Technology of Sensitive Data based on Artificial Intelligence (Liu et al., 2020)	Develop a sensitive data leakage prevention technology that can detect sensitive data based on a prefabricated security policy and self-learning artificial intelligence algorithm	Difficulty in achieving high accuracy due to the complexity of data and the need for continuous updates of the security policy	A comprehensive and effective technology that can detect sensitive data and provide a uniform measure of document importance to promote security policy updates
Using NLP and Machine Learning to Detect Data Privacy Violations	Utilize Named Entity Recognition (NER) to detect data privacy violations using three prominent NLP tools, including NLTK and Stanford NER	Challenges with applicability of NER in the context of data privacy, specific focus on NLTK and Stanford NER	A critical dimension to the evolving field of data loss prevention, offering insights into the technical methodologies and ethical considerations in handling sensitive information
Analyzing the Efficacy of Microsoft Presidio in Identifying Social Security Numbers in Unstructured Text	Examine the use of Microsoft Presidio for SSN identification, focusing on legal precedents and efficacy	Lack of encompassing federal law governing data privacy, specific exploration of Microsoft Presidio	A nuanced perspective on cybersecurity, data privacy, and compliance with regulations like HIPAA, showcasing Microsoft Presidio's capabilities
A Privacy-Preserving Approach to Extraction of Personal Information through Automatic Annotation and Federated Learning	Introduce WikiPII, an automatically labeled dataset, and explore the use of BERT-based NER with federated learning for personal information extraction	Challenges posed by label noise in automatic annotation, specific focus on BERT-based NER model	Scalable and privacy-aware solution bridging state-of-the-art machine learning with practical organizational needs, focusing on automatic annotation and federated learning
Named Entity Recognition in Unstructured Medical Text Documents	Automate the de-identification of IME reports using NER with OpenNLP and spaCy, focusing on efficiency and accuracy	Limitations on data set size, specific focus on IME reports, and comparison between OpenNLP and spaCy	Significant contributions to healthcare data management and patient confidentiality, emphasizing the automation of PII detection and privacy preservation through NER with OpenNLP and spaCy

3 Research Methodology

Research methodology is a critical part of thesis, as it describes all the research methods and procedures we have followed and below is in in depth description of the research methodology in our thesis.

3.1 Data collection

Data collection or Dataset creation is a Important part of the thesis as its required to finetune the BERT model according to our specification and get the desired results:

- Dataset Creation:** Dataset consisted of 18lakh entries of finely labelled entities. The creation of the Dataset involved two steps, One was finding the format in which we wanted to create a dataset and then was to generate a dataset as we could not find one because PII data is not available on internet. A fake PII data generator code was developed using Faker library in python which created our data then we used AI tool to put that created data into a format which was suitable for BERT to be trained on. This finely labelled dataset of Non PII and PII data was used to train BERT model and achieve good accuracy.

A	B	C	D
Sentence #	Word	POS	Tag
Sentence: 1	Thousands	NNS	O
	of	IN	O
	demonstrators	NNS	O
	have	VBP	O
	marched	VBN	O
	through	IN	O
	London	NNP	B-geo
	to	TO	O
	protest	VB	O
	the	DT	O
	war	NN	O
	in	IN	O
	Iraq	NNP	B-geo
	and	CC	O
	demand	VB	O
	the	DT	O
	withdrawal	NN	O
	of	IN	O
	British	JJ	B-gpe
	troops	NNS	O
	from	IN	O
	that	DT	O
	country	NN	O
	.	.	O

- Data Preprocessing:** Once the dataset was created, we must perform number of tasks before we can use it to finetune our pretrained BERT model. The first thing is filling out empty spaces in the Sentence# column with their respective sentence number and then we will encode the sentence column with values which the simpletransformers library could interpret by using label encoder from sklearn module. After that we renamed the columns in the format which the simpletransformer library will understand. This is why data preprocessing is required because libraries like simpletransformer need the dataset to be in a certain format. Along with that we performed an overall dataset check to see if there were any missing values and filled them across. (Winastwan, 2022)
- Data Splitting:** In this phase we separated the whole data in train and test. We took two variables X and Y. We assigned the independent variables which would be the sentence id and words to X and the dependent variable “labels” to Y. Then we split the data into train and test using the ‘train_test_split’ function from scikit -learn

library, where the partition the feature set ‘X’ and target set ‘Y’ into training and testing using an 80-20 split.

3.2 Model Selection

3.2.1 Choice of BERT Model

Why did we choose BERT Model for Named Entity Recognition:

1. **Pretrained Knowledge:** BERT is a pre-trained model which has been trained over 2500 million words enabling it to understand the context and semantics of words with more accuracy.
2. **Bidirectional Context:** BERT considers the context of words from both left and right given more better understanding of the words meaning within an sentence.
3. **Finetuning Capability:** BERT can be fine-tuned on a specific task such as Named Entity Recognition which is what we are using in our case, which gives BERT a upper hand as it is able to leverage pretrained knowledge and adapt to the particularities of the given task.
4. **State of the Art Performance:** BERT has achieved very great results in various Natural language processing tasks such as NER. (Winastwan, 2022)

Weaknesses of BERT:

1. **Computational intensity:** BERT model are relatively large and consume computational resources which may be a challenge for deployment in certain resource restricted environments
2. **Finetuning Complexity:** The finetuning process requires very careful selection of hyperparameters and understanding of architecture which may be a challenge for few people.

3.2.2 Finetuning Approach

Finetuning the pretrained BERT model for Named entity recognition involves the following steps:

1. **Data preparation:** Organizing the NER Data in format which is compatible with BERT including tokenization and alignment of sub words.
2. **Model Selection:** In our case we have selected the ‘bert-base-cased’ as a specific BERT variant which reflects the balance between performance and computational requirements. BERT-base is a variant that has 12 transformer layers making it smaller and less intensive in computation as compared to the BERT-large variant which has 24 layers. This particular compact model provides good trade-off between accuracy and computational cost making it more suitable for NER tasks. (Marshall, 2020)

3.3 Training

1. **Hyperparameter Selection and tuning:** Hyperparameter tuning involves selecting the appropriate settings that would govern the training process like the Learning rate, Number of training epoch and Batch Size. In our model we have set the hyperparameters to the below configuration:

- a. **Learning rate:** It is set to $1e-4$ which controls the step size during the optimization process. A high learning rate may lead to overshoot the optimal solution.
 - b. **Number of Epoch:** We have trained our Model over 8 epoch means the model has been trained 8 times over the same dataset. If we train the model too much over the same dataset it may overfit it and generate false positives. So, we have trained our model 8 times over the same dataset to get good accuracy. (Amy, 2023)
 - c. **Batch size:** We have set our training and evaluation batch size to 32 which determines how many examples were processed simultaneously during training and evaluation. (Amy, 2023)
2. **Hardware and Software:** We have trained our model over Google collab which uses Google hardware. Later we have used VS Code IDE to write python code to create a windows application which imports the finetuned BERT model and provides an interface to upload a text file and then read its content and classify it and later determine whether the file is Sensitive PII, Non-Sensitive PII or NON PII.

3.4 Evaluation Metrics

Evaluation involves accessing the model's performance in test using specific metrics. In our thesis we have considered the below metrics:

- **Precision:** It measures how many were true positive detections among all positive predictions.
- **Recall:** Measures how many true positive predictions were made out of actual positives.
- **Evaluation Loss:** Represents the evaluation loss on validation or test data.
- **F1 score:** It is the harmonic mean of precision and recall.

Considering our situation where we have to determine the accuracy of a trained BERT model the evaluation metrics like precision, recall, evaluation loss, F1 score are important because these scores depict the effectiveness of the model. The Precision shows how many true positive detections did the model have, the Recall shows how many true positive predictions we actually made out of actual positives, eval loss shows evaluation loss on validation and test data and F1 score is used to measure the accuracy of the model. All of these metrics help us determine the effectiveness and accuracy of the trained model due to which we have selected these evaluation metrics.

3.5 Ethical Consideration

Privacy and Security: The dataset was created using 'faker' library in python due to which all the data within the dataset was fake and is not associated with any person. (Uppal, 2020) Also, the application which we have created after importing the finetuned BERT module is focused on maintaining confidentiality and data privacy was not exposing any PII information when its detected. The application will hide the content of PII files if it gets

detected and it will show the list of entities which were detected for Non PII file it will display the content as there is no sensitive data present in that file.

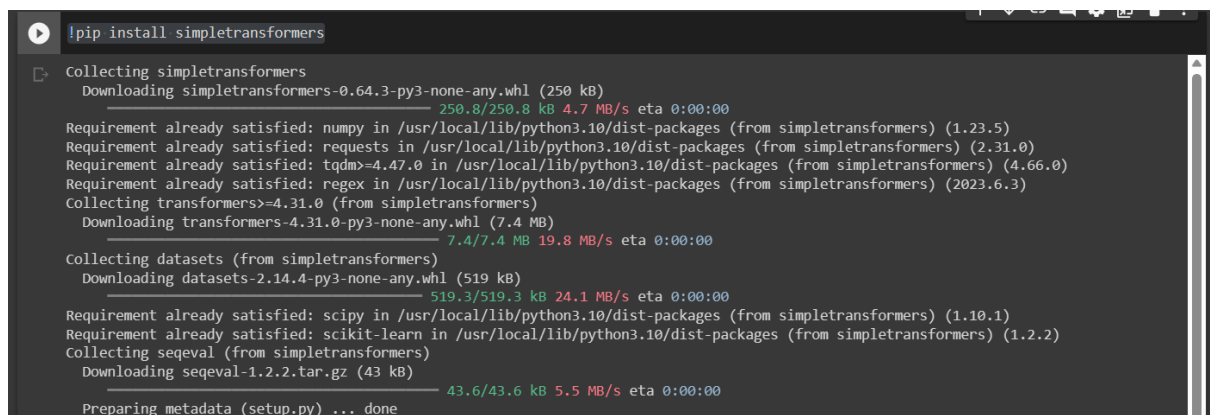
4 Implementation

Implementation is a severe part of the thesis where we have implemented our proposed work to achieve our goal. In this section I will in detail explain about the core details of implementation in the final stage of our development.

4.1 Training and finetuning of BERT model

BERT is a pretrained model which is used for various NLP tasks like Named entity recognition , it has been trained over dataset of 2600million words which makes it quite good and accurate for entity recognition, but we need to finetune the BERT model over our dataset in order to make it understand about what type of text or content we want to classify and what could be their labels and its needs to understand what we need it to classify in order to do that we need to perform finetuning of the BERT model using our custom dataset and below our main parts of the code which were important in the finetuning process.

1. Installing simpletransformers library



```
!pip install simpletransformers
Collecting simpletransformers
  Downloading simpletransformers-0.64.3-py3-none-any.whl (250 kB)
    250.8/250.8 kB 4.7 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from simpletransformers) (1.23.5)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from simpletransformers) (2.31.0)
Requirement already satisfied: tqdm>=4.47.0 in /usr/local/lib/python3.10/dist-packages (from simpletransformers) (4.66.0)
Requirement already satisfied: regex in /usr/local/lib/python3.10/dist-packages (from simpletransformers) (2023.6.3)
Collecting transformers>=4.31.0 (from simpletransformers)
  Downloading transformers-4.31.0-py3-none-any.whl (7.4 MB)
    7.4/7.4 MB 19.8 MB/s eta 0:00:00
Collecting datasets (from simpletransformers)
  Downloading datasets-2.14.4-py3-none-any.whl (519 kB)
    519.3/519.3 kB 24.1 MB/s eta 0:00:00
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from simpletransformers) (1.10.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from simpletransformers) (1.2.2)
Collecting sequeval (from simpletransformers)
  Downloading sequeval-1.2.2.tar.gz (43 kB)
    43.6/43.6 kB 5.5 MB/s eta 0:00:00
Preparing metadata (setup.py) ... done
```

This command is used to install the simpletransformers library which is thit'sost important library as this library provides a user-friendly interface for working with various transformers models including BERT, RoBERTa, GPT-2 and others and its particularly useful for NLP tasks such as NER and text classification. The library simplifies the process of training and using transformers model. It is built upon hugging face transformers library; Simple transformers give a simple and easy to use API for various NLP tasks. (Rajapakse, 2020)

2. Using pandas to import dataset.



```
[ ] import pandas as pd
data = pd.read_csv("Combinedtest.csv",encoding="latin1" )
```

Here we import the panda's library and give it a alias called pd. Pandas is widely used for data manipulation and analysis. Then we use pandas to read CSV file which is our dataset and store the data in a single dataframe object called "data". The encoding "latin1"

specifies the character encoding used to read the file. So now we have imported our dataset and have saved it in a dataframe. (Leong, 2021)

3. Dividing the dataset into test and train

```
[ ] data["labels"] = data["labels"].str.upper()

[ ] X= data[["sentence_id","words"]]
    Y =data["labels"]

[ ] x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size =0.2)

[ ] #building up train data and test data
    train_data = pd.DataFrame({"sentence_id":x_train["sentence_id"],"words":x_train["words"],"labels":y_train})
    test_data = pd.DataFrame({"sentence_id":x_test["sentence_id"],"words":x_test["words"],"labels":y_test})

[ ] train_data
```

In this code snippet the First line is where we converted the labels in upper case by using str.upper() function because it's best to convert it to upper case as its suitable for model training.

The second code box represents the differentiation of Independent and dependent variables into X and Y. X contains the independent variables sentence id and words while Y contains the dependent variable Label.

The third box represents splitting of test and train data into a ratio of 80:20. Where the test size is determined to be 20.

Then we build the train and test data, and we define the two dataframes 'train_data' and 'test_data' which can be used to train and evaluate the BERT model by passing them to appropriate function in Simple transformers library.

And then we train the data.

4. Importing essential functions from simple transformers

```
from simpletransformers.ner import NERModel,NERArgs

[ ] label = data["labels"].unique().tolist()
    label

['O',
 'B-GEO',
 'B-GPE',
 'NAME(PII)',
 'I-GEO',
 'B-ORG',
 'I-ORG',
 'B-TIM',
 'B-ART',
 'I-ART',
 'I-GPE',
 'I-TIM',
 'B-NAT',
 'B-EVE',
 'I-EVE',
 'I-NAT',
 'BIRTHDATE(PII)',
 'SOCIAL SECURITY NUMBER(PII)',
 'SSN(PII)']
```

First, we import the NERmodel and NERArgs from the simpletransformers library where NERModel is used to import specific NER models like BERT, GPT-2 and NERArgs is used to specify the hyperparameters we will be using for training purposes. Then we make a list of unique labels of the dataset which we want our model to classify for us.

5. Hyperparameter configuration

```
args = NERArgs()
args.num_train_epochs = 8
args.learning_rate = 1e-4
args.overwrite_output_dir = True
args.train_batch_size = 32
args.eval_batch_size = 32

[ ] model = NERModel('bert', 'bert-base-cased', labels=label, args =args)
```

In the first block we define the hyperparameters like number of epoch, batch size, learning rate which are quite essential in training the model effectively. The details about hyperparameters are covered in the Methodology section.

In the second block we import a specific BERT model named ‘base-base-cased’ which is a lightweight BERT model which works on 12 transformers and is a promising model which gives good results and uses less computational power.

6. Model training

```
model.train_model(train_data, eval_data = test_data, acc=accuracy_score)

/usr/local/lib/python3.10/dist-packages/simpletransformers/ner/ner_utils.py:190: FutureWarning: In a future version of pandas, a length 1 tuple will be returned
return [
100% ██████████ 2/2 [01:11<00:00, 35.15s/it]

Epoch 3 of 3: 100% ██████████ 3/3 [19:32<00:00, 395.72s/it]
Epochs 0/3. Running Loss: 0.0605. 100% ██████████ 1563/1563 [05:52<00:00, 4.97it/s]
/usr/local/lib/python3.10/dist-packages/torch/optim/lr_scheduler.py:139: UserWarning: Detected call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch it is preferred to call the `optimizer.step()` before `lr_scheduler.step()`. This might change the meaning of `step()` after upcoming versions.
warnings.warn("Detected call of `lr_scheduler.step()` before `optimizer.step()`.")
Epochs 1/3. Running Loss: 0.0487. 100% ██████████ 1563/1563 [06:12<00:00, 4.96it/s]
Epochs 2/3. Running Loss: 0.0785. 100% ██████████ 1563/1563 [06:15<00:00, 4.96it/s]
(4689, 0.0856593871053398)

[ ] result, model_outputs, preds_list = model.eval_model(test_data)

/usr/local/lib/python3.10/dist-packages/simpletransformers/ner/ner_utils.py:190: FutureWarning: In a future version of pandas, a length 1 tuple will be returned
return [
100% ██████████ 2/2 [00:24<00:00, 12.20s/it]

Running Evaluation: 100% ██████████ 1560/1560 [03:26<00:00, 4.51it/s]
/usr/local/lib/python3.10/dist-packages/sequeval/metrics/sequence_labeling.py:171: UserWarning: BIRTHDATE(PII) seems not to be NE tag.
warnings.warn("{} seems not to be NE tag.".format(chunk))
/usr/local/lib/python3.10/dist-packages/sequeval/metrics/sequence_labeling.py:171: UserWarning: NAME(PII) seems not to be NE tag.
warnings.warn("{} seems not to be NE tag.".format(chunk))
/usr/local/lib/python3.10/dist-packages/sequeval/metrics/sequence_labeling.py:171: UserWarning: SOCIAL SECURITY NUMBER(PII) seems not to be NE tag.
warnings.warn("{} seems not to be NE tag.".format(chunk))
/usr/local/lib/python3.10/dist-packages/sequeval/metrics/sequence_labeling.py:171: UserWarning: SSN(PII) seems not to be NE tag.
warnings.warn("{} seems not to be NE tag.".format(chunk))
```

In the first block used:

- **model.train_model(train_data)** to train the NER model on the provided train_data dataframe .
- **eval_data=test_data** to specify a dataset on which models performance will be evaluated during training.
- **acc = accuracy_score** to which is an argument to specify the custom accuracy function, this accuracy_score is typically used for NER tasks.

In the second block we evaluate the model where we used:

- **model.eval_model(test_data)** method to evaluate the trained NER model on test_data dataframe which will contain the testing features and labels.
- **result** is used to store evaluation metrics like recall, precision and F1 score.
- **Model_outputs** variable contains the raw output from the model.
- **Preds_list** variable contains the model’s final predictions for each instance in test data.

7. Result

```
result
{'eval_loss': 0.18471717099327856,
 'precision': 0.9161669843594178,
 'recall': 0.9011622131717493,
 'f1_score': 0.9086026553491954}
```

The result command shows the evaluation of metrics such as eval_loss, precision, recall and f1 score which are used to determine the accuracy and the effectiveness of the trained model. In our case we have got precision of 91.61% which is a good score for detecting true positives in overall.

8. Prediction

```
[ ] prediction, model_output = model.predict(["My name is Irene Adler i am from London and my birthdate is 07/30/1997 and SSN is 444-55-333"])
100% ██████████ 1/1 [00:00<00:00, 5.54it/s]
Running Prediction: 100% ██████████ 1/1 [00:00<00:00, 13.67it/s]

[ ] prediction
[[{'My': 'O'},
 {'name': 'O'},
 {'is': 'O'},
 {'Irene': 'NAME(PII)'},
 {'Adler': 'NAME(PII)'},
 {'i': 'O'},
 {'am': 'O'},
 {'from': 'O'},
 {'London': 'B-GEO'},
 {'and': 'O'},
 {'my': 'O'},
 {'birthdate': 'O'},
 {'is': 'O'},
 {'07/30/1997': 'BIRTHDATE(PII)'},
 {'and': 'O'},
 {'SSN': 'SOCIAL SECURITY NUMBER(PII)'},
 {'is': 'O'},
 {'444-55-333': 'SSN(PII)'}]]
```

In the prediction section we pass the prediction text using model.predict function and now the Bert model is able to classify each word of the sentence with a specific PII label which is our end goal. And by using the prediction command we can see that

9. Importing the finetuned BERT model and tokenizer

```
from transformers import BertTokenizer
import os
import shutil

# Instantiate your tokenizer (replace 'bert-base-uncased' with your tokenizer if different)
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Save your tokenizer
save_dir = "tokenizer_dir"
if not os.path.exists(save_dir):
    os.makedirs(save_dir)

tokenizer.save_pretrained(save_dir)

# Zip the directory
shutil.make_archive(save_dir, 'zip', save_dir)

# Now you can download your zip file from the Colab using the left side pane.
# Navigate to Files, refresh if you don't see your file and right click and download.

Downloading (...)solve/main/vocab.txt: 100% ██████████ 232k/232k [00:00<00:00, 3.41MB/s]
Downloading (...)tokenizer_config.json: 100% ██████████ 28.0/28.0 [00:00<00:00, 639B/s]
Downloading (...)ve/main/config.json: 100% ██████████ 570/570 [00:00<00:00, 15.8kB/s]
'/content/tokenizer_dir.zip'

[ ] model.save_model("/content/my_bert_model")

[ ] model.model.save_pretrained("/content/my_bert_model")

[ ] !zip -r /content/my_bert_model.zip /content/my_bert_model
```

After that we import or download the BERT model at our local machine along with the tokenizer which will be later imported in application to classify the PII and Non PII files.

4.2 Creating a Python Application using our final trained model

1. Importing the Important libraries and loading the saved model and tokenizer

```

import tkinter as tk
from tkinter import scrolledtext
from tkinter import ttk
from PIL import Image, ImageTk
from tkinter import filedialog, messagebox, scrolledtext
from transformers import AutoModelForTokenClassification, BertTokenizer
import torch
import pandas as pd
import re
from pandasgui import show

# Load Pretrained BERT NER Model
model_path = 'E:\\Ner\\my_bert_model (1)\\content\\my_bert_model'
model = AutoModelForTokenClassification.from_pretrained(model_path)

# Load the tokenizer
tokenizer_path = 'E:\\Ner\\Token' # Update this to the path of your vocab.txt
tokenizer = BertTokenizer.from_pretrained(tokenizer_path)

```

In the above code snippet we import all the necessary libraries required to create an application which will accept a .txt file from us as an input and will be able to classify, read, and classify whether the file is Sensitive PII, Non Sensitive PII, or Non PII.

We also load the final model we saved on our local system using `model_path` and we load the necessary tokenizer files like `vocab.txt` using the `tokenizer_path` function.

2. Function to predict the named entities

```

# Function for Predicting Named Entities
def predict_ner(text):
    inputs = tokenizer(text.split(), is_split_into_words=True, return_tensors="pt", truncation=True, padding=True)
    outputs = model(**inputs)
    predictions = torch.argmax(outputs.logits, dim=2)

    # Convert predicted token ids to labels
    tokens = tokenizer.convert_ids_to_tokens(inputs["input_ids"][0])
    new_labels = []
    for token, prediction in zip(tokens, predictions[0].tolist()):
        if token != tokenizer.pad_token:
            new_labels.append((token, model.config.id2label[prediction]))

    # Output predicted labels
    return new_labels

```

Here we use the `predict_ner` function that takes input from us and returns the predicted named entities using our finetuned BERT model. This function consists of the process of predicting named entities for given text input or text file using our model. It handles all the necessary steps of tokenization, model inference, and conversion of predictions to human-readable labels. It gives us named entities as a list of tuples to make it suitable for further analysis.

3. Load file function and logic behind classification

```

# Function to open a file dialog and load the file
def load_file():
    filename = filedialog.askopenfilename(filetypes=(("Text files", "*.txt"), ("all files", "*.*")))
    if filename:
        with open(filename, 'r') as file:
            text = file.read()
            ner_results = predict_ner(text)

            non_sensitive_pii_entities = ["NAME(PII)", "BIRTHDATE(PII)", "B-GEO"]
            sensitive_pii_entities = ["SSN(PII)", "SOCIAL SECURITY NUMBER(PII)"]

            pii_detected = [tag for _, tag in ner_results if tag in non_sensitive_pii_entities + sensitive_pii_entities]

            detected_non_sensitive_pii = any(tag in non_sensitive_pii_entities for tag in pii_detected)
            detected_sensitive_pii = any(tag in sensitive_pii_entities for tag in pii_detected)

```



```

if detected_sensitive_pii:
    messagebox.showwarning("Alert!!! Sensitive PII file detected", "Sensitive PII file detected. Please review the contents.")
elif detected_non_sensitive_pii:
    messagebox.showinfo("Non Sensitive PII file detected", "Non Sensitive PII file detected. Please review the contents.")

if detected_sensitive_pii or detected_non_sensitive_pii:
    detected_entities_str = ", ".join(set(pii_detected)) # List of unique detected PII entities
    output_text.delete('1.0', tk.END) # Clear the text widget before displaying new data
    output_text.insert(tk.END, "PII File Detected, Cannot disclose the confidential content\n")
    output_text.insert(tk.END, "Detected Entities: " + detected_entities_str + "\n")
elif len(pii_detected) == 0:
    messagebox.showinfo("No PII Detected", "This file is non-PII.")

# Create DataFrame and display it in a new window
df = pd.DataFrame({
    'Text': [token for token, _ in ner_results],
    'Tag': [tag for _, tag in ner_results]
})
show(df)

# Display the file content and the classification of each word
output_text.delete('1.0', tk.END) # Clear the text widget before displaying new data
output_text.insert(tk.END, "File Contents:\n\n" + text + "\n\nEntity Classification:\n\n")
formatted_results = str([f"{token: tag}" for token, tag in ner_results])
output_text.insert(tk.END, "[" + formatted_results + "]" + "\n")

```

In this code snippet we define the load file function where we accept filetype .txt so the user will be able to upload the txt and the logic behind the PII detection and PII alerts and handling. This codes provides an interactive way for users to load txt file and then apply named entity recognition, categorization of detected entities as sensitive and non – sensitive PII and appropriately handle and display the results by alerting the user what type of file it is.

In the code we have defined which entities are considered under Sensitive PII and non sensitive PII and have altered different alerts for each of the classification . To summarize if we the loaded txt file will contain entities which have been marked as Sensitive or Non sensitive PII it will give the respective alert and hide the content of the file in the output section to maintain data privacy and confidentiality and will only show what entities were detected and in a case where there were no PII entities detected it will show the content of the file and give alert that the file does not contain PII information.

4. Basic GUI Creation

```

# Create Basic GUI

window = tk.Tk()
window.title('Classification of PII and Non-PII Files for DLP')
window.geometry('1920x1080') # Specify the starting window size

# Load the background image
bg_image_path = 'E:\Ner\hacker3.jpg'
bg_image = Image.open(bg_image_path)
bg_image = bg_image.resize((1920, 1080), Image.ANTIALIAS) # Resize the image to fit the window
bg_image_tk = ImageTk.PhotoImage(bg_image)

# Set the background image
bg_label = tk.Label(window, image=bg_image_tk)
bg_label.place(relwidth=1, relheight=1)

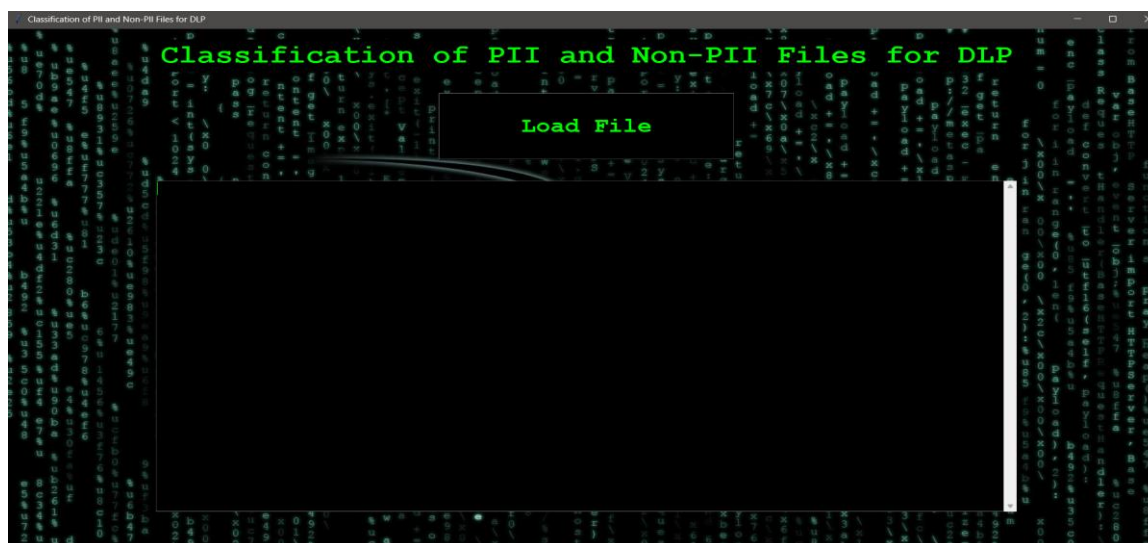
# Title Label
title_label = tk.Label(window, text='Classification of PII and Non-PII Files for DLP',
                        font=('Courier New', 30, 'bold'), fg='#00FF00', bg='black')
title_label.pack(pady=20)

# Load Button with hacker-like effect
load_button = tk.Button(window, text='Load File', command=load_file,
                        font=('Courier New', 24, 'bold'), fg='#00FF00', bg='black',
                        activeforeground='red', activebackground='black', width=20, height=2)

```

The above code created a visually appealing and Interactive GUI for classification of files into PII and Non PII categories. I have made the GUI simple and have given it a hacker effect by importing background image and other stuff. Here we have defined components of GUI like the load button the scrollable text area where the output will be produced.

5. Final Application View



This is how our final application looks like.

5 Evaluation and Test Cases

Evaluation of the Trained model is nothing but measuring accuracy, precision and recall and F1 score in terms of my thesis.

```
[28] result
{'eval_loss': 0.18471717099327856,
 'precision': 0.9161669843594178,
 'recall': 0.9011622131717493,
 'f1_score': 0.9086026553491954}
```

In my scenario we can evaluate the performance of the finetuned BERT model based on the above 4 factors which are described in detail below:

Evaluation loss: The eval_loss for our model is 0.18471 which represents the evaluation loss on validation or test data. A lower loss represent indicates that the model is performing well.

Precision: The precision for our model is 0.9161 which means 91.61%. It indicates how many true positive predictions were made of all positive predictions. High prediction rate always means that number of false positives will be less.

Recall: The recall of our model is 0.9011 or 90.11% which is used to measure how many true positive predictions were made from all actual positives. High recall means model is capturing most of the relevant information.

F1 Score: The F1 score in our case is 90.86% which is the harmonic mean of precision and recall. A High F1 score indicates model is well balanced.

As described in the introduction the real-world problem which I faced by accuracy of the DLP system to classify the files and detection, we were getting around 75% false positives alerts to improve the detection we can use our model to detect the PII files with more accuracy.

Test Cases:

There will be 3 major test cases for Checking if our model is working fine or not.

- Loading a file with Sensitive PII contents in it and checking whether its detecting or not.
- Loading a File with Non-Sensitive PII contents in it and testing.
- Loading a File with Non PII data and testing.

Each of the test case will be covered in the Configuration manual along with their respective results and screenshots.

6 Conclusion and Future work

6.1 Conclusion (Summary of findings)

I have successfully built two things in my entire project. One would be the building of the classifier model and secondly importing that model into a basic application to test out the functionality and the capabilities of our model.

Overall, I have successfully finetuned the Pre-trained BERT model over a dataset of 18lakh entries which were also created by me as the dataset was not publicly available. The final output of the model has an accuracy of more than 91% and is able to classify the words which is given to it as an input. It will tag each word with a classifier and label from which we would be able to understand whether its PII or Non PII. It works with great accuracy over the textual input. The PII entities which I have defined and trained the model are Name(PII), Birthdate(PII), Social Security Number(PII), SSN(PII) and B-geo(it indicates a geographical location).

Further I have imported that finetuned model into my local machine and use it to build a basic program or application which gives the user a load button option where it accepts .txt files. Once the file is submitted it will instantly read the content of the file and determine whether it has PII data or not. In the application I have categorized the PII into two groups Sensitive PII and Non-Sensitive PII where Sensitive PII means social security number and its keyword “SSN” whereas in Non-Sensitive PII there are entities like Name, Geo location, Birthdate etc. If the loaded .txt file would contain the Sensitive or Non sensitive PII data, it will generate respective alert which will notify the User about the content of the file and will display only the entities which were detected in the file, it wont display content of the file in order to ensure data privacy but it will alert the User that it may contain some Sensitive information. Also, if there are no PII entities in the file it will give an alert to the user notifying him/her that the file does not contain PII information and will display the content of the file as there is no sensitive data in it.

Overall, we have finetuned a pre trained BERT model using custom dataset for performing Named entity recognition to Classify and detect PII and Non PII files.

6.2 Impacts on the Field

I had proposed this topic to tackle a real-world problem faced by many cyber security teams in multiple countries which was the effectiveness, accuracy and approach of DLP tools to

detect PII files. As mentioned above most of these tools used keyword based and regular expression-based classification due to which they are unable to detect or classify entities like Name, Geo location and most the organization don't put focus on Non-Sensitive PII because according to them its important to protect from exfiltration of Sensitive PII files. But in my approach, we have explained that if Non-Sensitive PII are tagged along with Sensitive PII then there could be even more damage to the organization. Our model if integrated with DLP system would provide a more accurate and efficient way to track the PII files which are in motion or at rest. If we could create a machine learning based policy which would integrate our trained model, then we would be able to produce more accurate results and improve the quality of detection in DLP tool and get more true positive detection which would an great improve over the Legacy DLP rule based detection, which would in turn improve the quality of the Cyber security posture.

6.3 Limitation

Limitations of our projects are stated as follows:

- **Data Specificity :** We have finetuned the BERT model over a limited dataset which just trains the model to classify 4-5 PII entities , so it wont provide full proff model which would be able to detect more types of PII entities, we would require a massive dataset which could cover all the types of PII entities.
- **Model Constraints:** For training a bert model on a massive dataset would require more computational power which is hard to get in a resource restricted environment.
- **Ethical Considerations:** As we are trying to detect PII files , we need a massive PII dataset which is impossible to get due to data privacy laws in EU, so like I did we have to design a custom dataset like we did, but it wont be that accurate.

6.4 Future Work

Considering the current accuracy of the model and the type of dataset we have used to finetune it there could be number of things we could improve or enhance in the future.

1. **Quality of Dataset:** Currently our dataset is able to train the model to detect 5 types of PII entities, in future we can create a more massive dataset which would be able to detect most of the PII entities which would be more efficient and more applicable to solve real life problems.
2. **Integration with DLP Environment:** We could integrate our finetuned model with a working DLP tool and test it in the environment to check whether its producing promising results. Currently all the DLP solutions are enterprise level and we cannot test it in that environment due to limitation and availability.
3. **Improving accuracy:** If we could get a huge quality dataset with proper labels and unique data, we could finetune the model to gain more good results and accurate detections.
4. **Enhancement in the Application:** Our current application is only able to read and classify between .txt files so we add extra support where it will be able to read and classify from more types of files.

References

- Bernstein, C. (2023a) What is pii (personally identifiable information)?: Definition from TechTarget, Security. Available at: [https://www.techtarget.com/searchsecurity/definition/personally-identifiable-information-Pii#:~:text=Personally%20identifiable%20information%20\(PII\)%20is,anonymous%20data%20is%20considered%20PII.](https://www.techtarget.com/searchsecurity/definition/personally-identifiable-information-Pii#:~:text=Personally%20identifiable%20information%20(PII)%20is,anonymous%20data%20is%20considered%20PII.) (Accessed: 14 August 2023).
- Clarke, I. (2023) The troubles with legacy DLP Tools & How to solve them: Proofpoint us, Proofpoint. Available at: <https://www.proofpoint.com/us/blog/information-protection/troubles-with-legacy-dlp-tools-how-to-solve> (Accessed: 14 August 2023).
- Pearson, C., Seliya, N. and Dave, R. (2021) Named entity recognition in Unstructured Medical Text Documents | IEEE ... Available at: <https://ieeexplore.ieee.org/abstract/document/9698694> (Accessed: 13 August 2023).
- Winastwan, R. (2022) Named entity recognition with Bert in pytorch, Medium. Available at: <https://towardsdatascience.com/named-entity-recognition-with-bert-in-pytorch-a454405e0b6a> (Accessed: 14 August 2023).
- Horev, R. (2018) Bert explained: State of the art language model for NLP, Medium. Available at: <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270> (Accessed: 14 August 2023).
- Silva, P. et al. (2020) Using NLP and machine learning to detect data privacy violations | IEEE ..., Using NLP and Machine Learning to Detect Data Privacy Violations. Available at: <https://ieeexplore.ieee.org/abstract/document/9162683> (Accessed: 14 August 2023).
- Friebely, A. (2022) Analyzing the Efficacy of Microsoft Presidio in Identifying Social Security Numbers in Unstructured Text. Available at: <https://www.proquest.com/openview/31254af0453136664db6291485242df8/1?pq-origsite=gscholar&cbl=18750&diss=y> (Accessed: 14 August 2023).
- Cunningham, P. et al. (2022) Getting comfortable with data loss prevention policies in office 365, Practical 365. Available at: <https://practical365.com/getting-comfortable-data-loss-prevention-policies-office-365/> (Accessed: 14 August 2023).
- Guha, A. et al. (2021) A deep learning model for information loss prevention from Multi-Page ..., A Deep Learning Model for Information Loss Prevention From Multi-Page Digital Documents. Available at: <https://ieeexplore.ieee.org/document/9443107> (Accessed: 14 August 2023).
- Liu, D. et al. (2020) Research on leakage prevention technology of sensitive data based on ..., Research on Leakage Prevention Technology of Sensitive Data based on Artificial Intelligence. Available at: <https://ieeexplore.ieee.org/document/9152286/> (Accessed: 14 August 2023).

- Park, Y. (2010) IBM Research Report - dominoweb.draco.res.ibm.com. Available at: <https://dominoweb.draco.res.ibm.com/reports/rc25055.pdf> (Accessed: 14 August 2023)
- Hathurusinghe, R., Nejadgholi, I. and Bolic, M. (2021) A privacy-preserving approach to extraction of personal information through automatic annotation and Federated Learning, arXiv.org. Available at: <https://arxiv.org/abs/2105.09198> (Accessed: 14 August 2023).
- uppal, shaurya (2020) Python faker library, GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/python-faker-library/> (Accessed: 14 August 2023).
- @GrabNGoInfo, A. (2023) Hyperparameter tuning for Bertopic model in Python, Medium. Available at: [https://medium.com/grabngoinfo/hyperparameter-tuning-for-bertopic-model-in-python-104445778347#:~:text=Hyperparameter%20tuning%20is%20an%20important,NLP%20\(Natural%20Language%20Processing\).](https://medium.com/grabngoinfo/hyperparameter-tuning-for-bertopic-model-in-python-104445778347#:~:text=Hyperparameter%20tuning%20is%20an%20important,NLP%20(Natural%20Language%20Processing).) (Accessed: 14 August 2023).
- Rajapakse, T. (2020) Simple transformers-introducing the easiest bert, Roberta, xlnet, and XLM library., Medium. Available at: <https://towardsdatascience.com/simple-transformers-introducing-the-easiest-bert-roberta-xlnet-and-xlm-library-58bf8c59b2a3> (Accessed: 14 August 2023).
- Leong, N. (2021) Python for data science-A guide to pandas, Medium. Available at: <https://towardsdatascience.com/python-for-data-science-basics-of-pandas-5f8d9680617e> (Accessed: 14 August 2023).
- Marshall, C. (2020) What is named entity recognition (NER) and how can I use it?, Medium. Available at: <https://medium.com/mysuperaai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d> (Accessed: 14 August 2023).