Securing Sensitive Personal Data: Enhancing PII Security with AI/ML

Shuheng Pi, Jifei Huang, Xiangxiang Xie, Zihao Wang

Abstract

This paper presents an Artifical Intelligence(AI) framework to secure personally identifiable information (PII). We proposed a 2-layer approach designed to identify and anonymize PII within the specialized domain of the education sector. The framework can be divided into two components, the PII detection module and PII anonymization module. For PII detection module, the framework employs Named Entity Recognition (NER) model for PII token classification. Five Pre-trained Langauage Models (PLMs) are examined through fine-tuning with a dataset containing 6,807 student essays from a Kaggle provided by The Learning Agency Lab [1]. We conducted a comparative experiment to evaluate the model performance before and after fine-tuning. Our experimental findings indicate that the Fine-tuned RoBERTa model achieves the highest F1 score of 95.78%. For the PII anonymization module, we propose four masking functions, allowing users to select either reversible or irreversible PII masking based on their specific business needs. Finally, we conclude the limitations and future work of our framework. GitHub Link: https://github.com/seanpsh/bt5153_gp

1. Introduction

In this digital era, considering data as the "new oil" emphasizes its value and the risks associated with mishandling it. Concerns over data breaches stand out, causing reputational, financial, ransomware-related losses, and business disruptions. For instance, U.S. data breaches average a cost of $8.19 million [2]. Companies also deal with indirect costs like compensating impacted customers. The first six months of 2022 marked a 33% rise in significant data breaches in Australia, affecting over 40% of the people. Firms violating privacy regulations could face fines reaching $50 million, triple the illicit profit from data misuse, or 30% of their annual turnover [3]. Hence, the importance of securing sensitive personal data cannot be overstated.

Traditional manual reviews are thorough yet impractical due to their laborious nature and the vast amount of data involved. This project introduces an PLM-powered approach to PII security, proposing an efficient system for the identification and anonymization of sensitive information within text.

1.1. Business Insights

Incorporating AI/ML into PII security brings various business benefits, including enhanced trust and growth, improved risk management, and increased data utility, positioning organizations for success in a data-driven landscape.

- **Trust and Business Growth:** Securing PII enhances trust among consumers and partners, vital for growth in privacy-focused sectors. TeraDact Solutions uses Amazon Comprehend for advanced Natural Language Processing (NLP)-based PII redaction, enhancing its security and customer reach, thereby supporting business growth [4].
- **Risk Management and Compliance:** Implementing PII detection frameworks is crucial for ensuring compliance with data protection laws such as the General Data Protection Regulation (GDPR) by effectively addressing real-time threats, thus preventing data breaches and avoiding financial penalties. They are vital for managing privacy risks related to the collection, transfer, storage, and processing of personal data [5].
- **Enhancing Data Utility:** Effective PII management allows healthcare providers to leverage large datasets for advancements in treatments and operations without risking patient confidentiality. Techniques like k-anonymity and l-diversity modify PII in such a way that individual identities are protected while the data remains detailed enough for meaningful analysis [6].

1.2. Contributions

In this paper, we present our contributions in four key areas:

1. **Exploration of PLMs:** We explore five pre-trained language models tailored for NER tasks, including BERT, DistilBERT, DeBERTa, ALBERT, and RoBERTa, and obtain their model performance results by using evaluation metrics such as cross-entropy loss, recall, precision, and F1 score.

2. **Fine-tuning of PLMs:** We fine-tune these pre-trained models with Kaggle dataset to further boost their performance and gather their respective performance metrics as outlined previously.

3. **Model Comparison and Selection:** We identify the best-performing model by comparing the results of these pre-trained models before and after fine-tuning for our PII detection task.

4. **PII Protection Framework Construction:** We propose a effective and flexible framework for PII detection and anonymization, which incorporates PII detection function and anonymization functions for PII entities masking, hashing, and removing sensitive information. This framework ensures the protection of privacy while maintaining data usability.

The paper is structured as follows: Section II outlines our methodology for PII protection, covering NER and PII masking. Section III describes the experimental design for PII detector selection, including experiment environmental setup, usage of python packages, fine-tuning dataset preparation, evaluation and comparison of fine-tuned PLMs models for PII detection. The development of a PII masking functions for data anonymizatio module is described in Section IV. Section V concludes with a discussion of findings, limitations, and potential future directions.

2. Methodology: PII Protection Framework

We propose a 2-layer module PII protection framework: an identification module employing NER to detect PII in text, and an anonymization module to mask the detected PII entities.

2.1. Pre-trained Language Models

PLMs play a pivotal role in the field of NLP by providing a strong foundation for a wide range of tasks. These models are initially trained on large amounts of text data in an unsupervised manner, such as conll2003 [7], BookCorpus [8], English Wikipedia etc, learning a deep understanding of language patterns and context. Notable examples of PLMs such as BERT, DistilBert, Deberta, Albert, and Roberta, which have consistently demonstrated exceptional performance across various language-related tasks compared to traditional machine learning models.

Given their ability to leverage vast amounts of training data, PLMs able to generalize well to tackle new NER challenges. As a result, we decide to use pre fine-tuning PLMs as the baseline models for our project, capitalizing on their adaptability to diverse NER tasks.

2.2. Named Entity Recognization

NER is an essential technique in NLP that involves detecting and categorizing key information entities like person names, organizations, locations, events, quantities and more in unstructured text [9]. Essentially, NER acts as a mechanism to pinpoint and extract the most pertinent pieces of information embedded within text, without the need for manual analysis.

In our project, we utilize NER models due to their significant advantages over traditional rule-based methods like some masking built-in functions in msticpy package for PII detection. Unlike traditional methods, which are constrained by fixed patterns and may overlook linguistic variations, NER leverages deep learning to adapt dynamically to diverse data contexts, thereby enhancing detection accuracy and robustness.

2.3. Fine-tuning

Fine-tuning serves to refine the capabilities of pre-trained models, customizing them for specific tasks like NER applications. By adjusting model parameters to suit the unique characteristics of PII data, fine-tuning enhances precision and minimizes errors in entity recognition, thus crucial for achieving high accuracy and specificity in identifying sensitive information. To meet the specific requirements of NER tasks, particularly in PII detection, we utilize PLMs such as BERT, DistilBERT, DeBERTa, ALBERT, and RoBERTa with further fine-tuning.

2.4. PII-Masking

Once PII is identified, masking techniques are employed to ensure data privacy. These techniques vary from replacing data with pseudonyms or placeholders to complete data anonymization. An example of PII Masking is illustrated in the table below. The table demonstrates the before and after effects of applying PII masking to text data. It shows the original data, its classification by a PII detector, and the final masked output where sensitive information is replaced with placeholders to protect data privacy.

Table 1. Example of PII Masking in Text Tokens

Tokens	**PII Detector Result**	**PII Masking**
`Peter`	B-Person	[Person Name]
`has`	O	has
`email`	O	email
`peter@abc.com`	B-Email	[Email]

3. Experiment Design for PII detection

3.1. Experiment Setup

The experiment is running on Kaggle platform with GPU P100, 16GB GPU, and 29GB RAM. In order to construct our models, we utilized python packages such as NumPy, Pandas, scikit-learn, Transformers and SeqEval. When evaluating the performance of detection models, cross entropy loss, F1, precision, and recall are selected to provide a more comprehensive analysis of performance results.

3.2. Dataset Selection

We selected the Kaggle dataset provided by The Learning Agency Lab with 6,807 essays designated for training and testing in our experiments [1]. Our choice was driven by several factors: Firstly, the dataset's large size provides a substantial sample for language model training. Secondly, the dataset features detailed PII labeling, enabling accurate entity recognition. Thirdly, its structured format reduces the necessity for manual cleaning and labeling, streaming our data preparation process. Lastly, this dataset focusing on an educational context narrows down our project scope, aiding in the development of models that effectively address PII concerns within academic environments and safeguard sensitive student information.

Each document within the dataset is formatted in JSON, comprising a unique identifier, the complete text of the essay, a sequence of tokens, details about trailing whitespace, and annotations for each token. Tokens are annotated using the BIO (Beginning, Inner, Outer) format, with "B-" marking the beginning of a PII entity, "I-" indicating continuation of an entity, and "O" representing non-PII tokens.

Key PII entities annotated within the dataset include:

1. **NAME_STUDENT**: Any full or partial names of students, excluding those of instructors or other individuals.
2. **EMAIL**: Email addresses associated with students.
3. **USERNAME**: Usernames used by students across different platforms.
4. **ID_NUM**: Numbers or character sequences that can uniquely identify a student, such as student IDs or social security numbers.
5. **PHONE_NUM**: Telephone numbers linked to students.
6. **URL_PERSONAL**: URLs that could potentially identify a student.
7. **STREET_ADDRESS**: Full or partial residential addresses of students.

3.3. Data Preprocessing

For our study, the well-structured Kaggle dataset required no data cleaning, such as removing duplicates or repairing damaged entries. However, we convert textual labels that categorize different types of PII in BIO format into numerical identifiers through dictionary mappings. This transformation streamlines the handling and processing of the data, making it more suitable for analysis by machine learning algorithms.

To further refine the data, the dataset undergoes tokenization and alignment using the Hugging Face transformers library, which is equipped with advanced NLP tools and PLMs. These steps include [10]:

- **Tokenization**: The text is broken down into smaller units or tokens. This process is crucial for the models to analyze and learn from the text effectively.
- **Label Alignment**: After tokenization, every token is aligned to an appropriate label. Proper alignment is vital for tasks like NER, where the model must accurately associate labels with the correct tokens, including handling subwords and special tokens.
- **Data Collation**: Standardizing input data lengths through batching and padding allows for more efficient processing during model training.
- **Train-Test Split**: The dataset is split into training and validation sets to assess PLMs capabilities and performance on new and unseen data.

The preprocessing measures undertaken—including converting textual labels to numerical identifiers, and refining data through tokenization and alignment—lay a solid foundation for our ML/AI models discussed in the methodology section. These steps optimize dataset preparation, ensuring precise training and enhancing the performance and reliability of our models in detecting and anonymizing PII within educational data.

3.4. Evaluation Measures

In evaluating the performance of PLMs for NER tasks, we employ a set of evaluation metrics, such as cross-entropy loss function, recall, precision, and F1 score. These metrics serve as benchmarks for assessing the model's effectiveness while minimizing False Positives (FP) and False Negatives (FN). Each metric offers unique insights into various aspects of the model's performance, enabling a comprehensive evaluation that balances accuracy and completeness in PII detection.

Cross-Entropy Loss: The token-level cross-entropy loss quantifies the difference between predicted and actual entity

type distributions for each token in a sequence, guiding the model to make more accurate predictions during training for tasks like named entity recognition.

$$CrossEntropyLoss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i} y_{it}\log(p_{it})$$

- N is the total number of samples
- T_i is the number of tokens in the i-th sample
- y_{it} is a binary indicator of whether token t in sample i belongs to an entity or not
- p_{it} is the predicted probability that token t in sample i belongs to an entity

Recall: It is the proportion of the number of samples correctly predicted to be positive by the classifier to the total number of samples actually positive.

$$Recall = \frac{TP}{TP + FN}$$

Precision: It is the proportion of the number of samples predicted as positive by the classifier that are actually correct.

$$Precision = \frac{TP}{TP + FP}$$

F1 Score: Both the precision and recall of the classifier are considered. The F1 score ranges from 0 to 1, with higher values indicating better performance of the classifier.

$$F1score = \frac{(1+\beta^2)\cdot \text{precision}\cdot \text{recall}}{\beta^2\cdot \text{precision} + \text{recall}}$$

- β is a parameter that adjusts the importance of recall relative to precision.

In PII detection, the choice of an appropriate evaluation metric is crucial due to the sensitivity of the data and the severe consequences of missing PII, such as privacy breaches and compliance issues. Given the high cost of false negatives, it is vital to emphasize recall. Therefore, an F1 score with a beta value of 5, which prioritizes recall over precision, is highly suitable for these tasks. Our evaluation uses a range of metrics, with the adjusted F1 score serving as the primary measure to identify the most effective model for PII detection.

3.5. Experiment: Performance analysis of Fine Tuned PII Detectors

3.5.1. Control Variables

To ensure a controlled and consistent environment for performance analysis across fine-tuned PLMs, we standardized critical hyperparameters in our experiments. The learning rate was set at 2.5e-5 across all models to promote gradual and stable learning, and ensure steady convergence. A consistent weight decay of 0.02 was applied to regularize the models and mitigate overfitting — a common challenge with complex models.

All models were trained for three epochs to mitigate the risks of underfitting and overfitting. This approach ensures that each model has fair learning without simply memorizing it. To capture the most subtle changes and ensure sensitivity to the training data, we employed minimal batch sizes. This granularity in the update process allows for precise adjustments to the models' weights.

Keeping above control variable constant enabled us to accurately attribute any observed enhancements in performance to the fine-tuning process. By adopting such a rigorous methodology, we ensured reliable comparisons and gained meaningful insights into the effectiveness of each fine-tuned PII detector candidate.

3.5.2. BERT

BERT stands for Bidirectional Encoder Representations from Transformers. It's a popular PLM developed by researchers at Google in 2018 [11].

Table 2. Pre-trained bert-base-uncased Model Fine Tune Performance

Epoch	Training Loss	Validation Loss	Recall	Precision	F1
Pre	-	2.5947	0.0338	0	0.0005
1	0.0041	0.0039	0.6351	0.5434	0.6310
2	0.0013	0.0026	0.7500	0.6727	0.7467
3	0.0003	0.0031	0.7770	0.7012	0.7738

Table 2 illustrates that directly applying the pre-trained BERT model to our task without fine-tuning yields unsatisfactory performance. This could be attributed to the fact that the model has only undergone unsupervised learning on large corpus text data, and its parameters are not optimized for our specific NER task. Without task-specific optimization, the model struggles to achieve accurate classification.

In contrast, after proceeding with fine-tuning on domain-specific dataset, we observe significant enhancements even after just a single epoch. Validation losses decrease noticeably, accompanied by substantial improvements in precision,

recall, and F1 score. While there is a slight increase in validation loss during the third epoch compared to the second, the overall trend still performs well, with high precision, recall, and F1 score steadily improving across epochs.

3.5.3. DISTILBERT

DistilBERT is a smaller version of the original BERT model, which was introduced by Hugging Face in 2019. It maintains approximately 97% of BERT's effectiveness in language understanding tasks but with 40% fewer parameters and 60% faster processing speeds [12].

Table 3. Pre-trained distilbert-base-uncased Model Fine Tune Performance

Epoch	Training Loss	Validation Loss	Recall	Precision	F1
Pre	-	2.9541	0.0270	0	0.0004
1	0.0038	0.0028	0.5574	0.6892	0.6830
2	0.0012	0.0024	0.6031	0.7905	0.7812
3	0.0002	0.0023	0.7250	0.7838	0.7813

Like BERT, during the pre-training phase (Epoch Pre), the model shows high validation loss and poor performance on recall, precision, and F1 score metrics.

Fine-tuning resulted in significant metric improvements, notably between pre-training and Epoch 1. However, from Epoch 2 to 3, improvements were less pronounced, suggesting the model's performance is stabilizing near its optimal capacity.

3.5.4. DEBERTA

DeBERTa is a variant of BERT, which is a popular pre-trained language representation model developed by Google. DeBERTa stands for "Decoding-enhanced BERT with disentangled attention", and it extends the original BERT model by introducing several enhancements to improve its performance on various natural language understanding tasks [13].

Table 4. Pre-trained deberta-v3-base Model Fine Tune Performance

Epoch	Training Loss	Validation Loss	Recall	Precision	F1
Pre	-	3.0298	0	0	0
1	0.0029	0.0021	0.6400	0.6621	0.6408
2	0.0014	0.0009	0.8733	0.7238	0.8664
3	0.0003	0.0008	0.8867	0.7917	0.8826

Same as prior cases, the pre-fine-tuning model shows poor performance in detecting PII with zero scores for recall, precision, and F1. However, after the 1st epoch of fine-tuning, a noticeable improvement is observed, followed by progressive gains. The model achieved the highest F1 score of 0.8826 in the 3rd epoch.

3.5.5. ALBERT

ALBERT (A Lite BERT) is a neural language representation model introduced by Google Research. It's designed to be a more efficient variant of BERT, which is a highly successful pre-trained model for natural language understanding tasks [14].

We utilize two specific configurations: the pre-trained albert-base-v2 model [14], which is trained unsupervised on a broad corpus to learn general language representations and serves as a foundation for various NLP tasks. The albert-base-v2-finetuned-ner model [15], which trained on the conll2003 dataset.

Table 5. Pre-trained albert-base-v2 Model Fine Tune Performance

Epoch	Training Loss	Validation Loss	Recall	Precision	F1
Pre	-	3.3478	0	0	0
1	0.0050	0.0019	0.6786	0.6597	0.6778
2	0.0007	0.0015	0.7929	0.6647	0.7870
3	0.0004	0.0013	0.7929	0.8043	0.7933

Table 5 shows the performance of the pre-trained albert-base-v2 model. The evaluation metrics indicated steady improvements after fine-tuning. In Epoch 3, it achieved a 0.8043 precision score, indicating a refined ability to accurately identify PII without over-classification.

Table 6. Pre-trained albert-base-v2-finetuned-ner Model Fine Tune Performance

Epoch	Training Loss	Validation Loss	Recall	Precision	F1
Pre	-	2.3605	0	0	0
1	0.0049	0.0045	0.5714	0.2857	0.5503
2	0.0015	0.0021	0.7071	0.7500	0.7087
3	0.0007	0.0015	0.7714	0.7826	0.7719

Table 6 presents the albert-base-v2-finetuned-ner model, which starts with the same pre-trained foundation but is further pre-fine-tuned specifically for NER tasks. Initial results also show no capability in detecting PII as well. Whereas, fine-tuning achieves significant advancements in all evaluation measures.

The comparison of Table 5 and Table 6 in our experiment reveals that the general albert-base-v2 model fine-tuned on broad corpora exhibits a higher F1 score than the albert-base-v2-finetuned-ner model. The reduced F1 score for the pre-fine-tuned NER model may result from a misalignment with the dataset's specific features or indicate a necessity for

further tuning of hyperparameters to enhance the model's detection efficacy.

3.5.6. RoBERTa

RoBERTa (Robustly Optimized BERT approach) represents an evolution of the BERT model that was introduced by Facebook AI in 2019 [16]. As an improved recipe for training BERT models, RoBERTa is able to match the performance of post-BERT methods with several modifications in the training process, such as training the model longer with bigger batches and removing the next sentence prediction objective.

Table 7. Pre-trained roberta-base Model Fine Tune Performance

Epoch	Training Loss	Validation Loss	Recall	Precision	F1
Pre	-	2.6344	0.0354	0	0.0005
1	0.0037	0.0024	0.6028	0.4670	0.5962
2	0.0017	0.0009	0.9432	0.8110	0.9374
3	0.0005	0.0010	0.9149	0.8487	0.9122

Like other PLMs, the performance of RoBERTa during pre-trained phase is poor. In epoch 2, the model achieved the highest F1 score of 0.9374. However, a reduction in F1 score from 0.9374 to 0.9122 in Epoch 3 indicated that overfitting is occurred.

3.5.7. Model Comparison and Selection

Table 8 compares the performance of all fine-tuned PLMs. Each row represents a specific model and its best fine-tuning results. The results shows that the RoBERTa model outperforms all others, achieving the highest F1 score of 0.9374 in its second epoch. Following closely is DeBERTa, which achieves the second-highest performance with an F1 score of 0.8826 in its third epoch. DistilBERT and ALBERT demonstrate a similar competitive performances.

An interesting finding is that the fine-tuned ALBERT model, which underwent pre-training on a custom corpus for NER tasks, performs less effectively compared to its base ALBERT model counterpart. This could be due to the base ALBERT model having stronger generalization abilities and being more adept at learning new specific tasks, while the fine-tuned ALBERT model may encounter difficulty in fully capturing the complexity of the task at hand. Consequently, this limitation contributes to a reduction in performance for the fine-tuned ALBERT model (F1=0.7719) compared to its base counterpart (F1=0.7933).

3.6. Further Enhancement on fine-tuned RoBERTa model

Based on Table 8's comparison result, the fine-tuned RoBERTa model achieved the highest F1 score and selected as the final PII detection in our proposed PII protection framework. However, before we move to Section IV, we aimed to further enhance the performance of this RoBERTa model. Due to limited size of training dataset, RoBERTa showed overfitting after the second epoch. To address this, we expand the training dataset by applying external text samples containing PII entities which are generated by GAI model. Such external text samples shared the same text types and PII categories as our original training dataset.

After combining with the external dataset, our dataset expanded from 6,807 to 11,241 text samples (we keep to use the same validation set as the previous experiments). Furthermore, in this further enhancement stage, we only fine tune the model with text samples containing PII entities in training set and drop those samples without PII entities to further enhance the detection capability of the Roberta model in short time consuming. The enhanced results are shown in the table 9, demonstrated continued performance improvement. Ultimately, RoBERTa achieved an F1 score of 0.9575.

4. PII Anonymization

4.1. Detection result pre-processing

In the subsequent section of our PII detection and anonymization framework, we will employ the fine-tuned Roberta as out PII detector, which demonstrated optimal performance in the comparative experimentation conducted earlier. Due to the PLM model's inherent constraint on the maximum token limit, the PII detector will process text samples of up to 512 tokens at a time. Upon receiving a text sample, the PII detector will first tokenize the text and then identify the label corresponding to each token. If a token's label falls within the category of PII tags, its position in the original sentence will be recorded for subsequent anonymization, the example is shown in Table 9. Notably, as the recorded PII tokens may not represent complete entities, they will be merged with the preceding entity and restored to their readable forms in the original sentence, facilitating the subsequent anonymization process.

4.2. Anonymization under encryption and decryption

Based on the varying levels of data privacy and confidentiality of documents and the potential decryption needs of the anonymized entities, we have implemented an efficient PII anonymization function based on encryption and decryption mechanisms. Firstly, the anonymization function encrypts the PII entities detected by the PII detector using a Secure

Table 8. Comparison of Results from Best Fine-Tuned Pre-trained Language Models

Model	Epoch	Training Loss	Validation Loss	Recall	Precision	F1
BERT	3	0.0003	0.0031	0.7770	0.7012	0.7738
DistilBERT	3	0.0002	0.0023	0.7250	0.7838	0.7813
DeBERTa	3	0.0003	0.0008	0.8867	0.7917	0.8826
ALBERT	3	0.0004	0.0013	0.7929	0.8043	0.7933
ALBERT NER	3	0.0007	0.0015	0.7714	0.7826	0.7719
RoBERTa	2	0.0017	0.0009	0.9432	0.8110	0.9374

Table 9. Performance of Further enhancement on fine-tuned Roberta Model

Epoch	Training Loss	Validation Loss	Recall	Precision	F1
1	0.0092	0.0019	0.9291	0.7528	0.9208
2	0.0057	0.0025	0.8298	0.7697	0.8208
3	0.0038	0.0018	0.9645	0.8095	0.9575

Hash Algorithm 256-bit (SHA-256) encryption mechanism. SHA-256 is a cryptographic hash function used to generate fixed-size hash values for data. It transforms input data into a 256-bit hash value, ideally producing a unique output for different inputs. SHA-256 is widely used in encryption, data integrity verification, and cryptography. For example, anonymizing 'peterjackson@gmail.com' as 'cadcb8a892ee00fb428b1bfdc5b26155d9c71e9ec315b906e6c836932826f7a1'.

Simultaneously, we generate a corresponding table of hash values for each PII entity, which will be managed by person with privilege permissions (such as the owner of the original document). This provides reversibility after PII anonymization. However, if irreversible anonymization is desired, we also provide three additional irreversible anonymization functions in the following sub-section for users to choose from according to their needs.

4.3. Additional Anonymization Methods

Our proposed PII detection & anonymization framework offers additional methods for PII anonymization to cater to various user groups and scenarios. The details of these three additional algorithms are provided in the appendix below.

4.3.1. Option 1: Anonymization under detection tag

Masking under the detection tag refers to the process of replacing PII entities (i.e., peterjackson@gmail.com) with their predicted PII tags (i.e., [EMAIL]). This method helps users understand the type of information that has been anonymized in the original text, such as email addresses or street addresses, when using the anonymized text or dataset.

Algorithm 1 PII Anonymization under encryption and decryption

Require: Stored PII Entities - a list of detected PII entities, OriginalText - the original text containing PII

```
# Sort entities based on their start positions
FormattedResults ← SortEntitiesByStart(Entities)
# Initialize offset
Offset ← 0
# Iterate through each entity
for all Result in Stored PII Entities do
    # Extract entity tag
    PII_entity ← ExtractEntityTag(PII_entity)
    sha256_hash ← hashlib.sha256()
    sha256_hash.update(PII_entity)
    txt_hash ← sha256_hash.hexdigest()

    # Calculate start and end positions considering the offset
    Start ← Result.Start + Offset
    End ← Result.End + Offset

    # Write masked data to CSV file
    Open en_file in append mode
    Create a CSV writer object
    Create a row with elements txt_hash, tag, org_txt
    Write the row to the CSV file
    Close the CSV file

    # Replace the entity with its tag in the original text
    MaskedText ← ReplaceEntityWithTag(OriginalText, [txt_hash], Start, End)

    # Update the offset
    Offset ← Offset + (Length(txt_hash) + 2) - (Result.End - Result.Start)
end for
# Return the masked text
MaskedText ← MaskedText
```

Table 10. Stored format of detected PII entities

Input Text: Kaur is a student, he is 18 years old. His email is Kaurkk@gmail.com. His website is https://www.bell-kelly.net/tag/tags/categoriesprivacy.htm. His dream is to become a football player.

No.	Entity	Start	End	word
1	NAME_STUDENT	0	4	Kaur
2	EMAIL	52	68	Kaurkk@gmail.com
3	URL_PERSONAL	85	142	https://www.bell-kelly.net/tag/tags/categoriesprivacy.htm

The table presents the stored format of detected PII entities. Each row represents a detected PII entity, including its identification number, entity type, start and end positions in the original text, and the corresponding word or phrase.

Algorithm 2 indicates the masking process.

4.3.2. Option 2: Anonymization under Redaction

Option 2 anonymizes all PII entities as [Redaction], no matter PII entities belong to which PII tag. For example, anonymizing 'peterjackson@gmail.com' as [Redaction]. Compared to option 1, this anonymization method is more thorough. Users of the anonymized document cannot discern the types or contents of the original PII entities through the masks. The anonymization under Redaction algorithm can be found in Algorithm 3.

4.3.3. Option 3: Anonymization under obfuscation

Option 3 anonymizes PII entities by replacing some of the letters with placeholders to achieve a obfuscated effect. For example, anonymizing "peterjackson@gmail.com" as "pxxxxxxxxxx@gmail.com". Users can freely choose anonymization methods based on their needs and considerations for the level of data privacy. The anonymization under obfuscation algorithm can be found in Algorithm 4.

5. Conclusion

In this study, we introduce a 2-layer AI/ML methodology for PII protection, integrating an NER-based identification with a PII anonymization module. The use of fine-tuned PLMs further boosts PII detection efficacy. Our evaluation of six PLMs — BERT, DistilBERT, DeBERTa, ALBERT, and RoBERTa—revealed that RoBERTa markedly outperformed the others, achieving an F1 score of 0.9374 by its second epoch. Interestingly, the fine-tuned ALBERT model showed lower performance than its generic counterpart, likely due to inadequate adaptation to the complexities of the PII detection task.

Upon detection of PII, we explore four PII masking techniques to ensure data privacy: reversible encryption, detection tagging, redaction, and obfuscation. The first technique permits decryption, while the remaining three involve irreversible encryption, rendering the data permanently encrypted. The selection of an masking technique is based on specific business requirements.

5.1. Limitations and Future Works

The limitations of the current PII detection model include restricted generalization capabilities due to limited PII tagging, which confines its applicability across different contexts or types of PII. To address this, training the model on a more diverse range of documents, incorporating various PII types, and integrating broader datasets such as the Enron dataset and resources from Lakera AI is planned.

Additionally, the model's proficiency is currently limited to data in a single language, potentially hindering its global applicability across various linguistic settings. To address this, exploring multilingual text corpora will enhance the model's performance across different languages and broaden its global usability.

References

[1] Kaggle. *PII Detection Removal from Educational Data*. Available online at: Kaggle Competition Dataset [Online; accessed 20-April-2024].

[2] Stryker Networks. *Potential Risks That Insider Threats Pose to PII*. Available online at: Stryker Networks News [Accessed 20-April-2024].

[3] LinkedIn. *Protecting Personally Identifiable Data (PII) is a Business Necessity*. Available online at: LinkedIn Article [Online; accessed 20-April-2024].

[4] Amazon Web Services. *Detecting and redacting PII using Amazon Comprehend*. Available online at: Amazon Web Services Blogs [Online; accessed 20-April-2024].

[5] Microsoft. *Securing AI and ML projects: Data and cyber risk management*. Available online at: Microsoft Industry [Online; accessed 20-April-2024].

[6] Iyiola E. Olatunji, Jens Rauch, Matthias Katzensteiner, and Megha Khosla. *A Review of Anonymization for Healthcare Data*. *Big Data*, Mary Ann Liebert Inc,

March 2022. DOI: 10.1089/big.2021.0169. ISSN: 2167-647X.

[7] Hugging Face. *conll2003 Datasets at Hugging Face* Available online at: Hugging Face [Online; accessed 20-April-2024].

[8] Zhu, Yukun and Kiros, Ryan and Zemel, Rich and Salakhutdinov, Ruslan and Urtasun, Raquel and Torralba, Antonio and Fidler, Sanja *Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books*. The IEEE International Conference on Computer Vision (ICCV).

[9] DataCamp. *What is Named Entity Recognition (NER)? Methods, Use Cases, and Challenges* Available online at: DataCamp[Online; accessed 30-April-2024].

[10] Hugging Face *Token classification at Hugging Face* Available online at: Hugging Face [Online; accessed 20-April-2024].

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv:1810.04805, 2019. Available online at: https://arxiv.org/abs/1810.04805.

[12] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: .

[13] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. arXiv:2006.03654, 2021. Available: https://arxiv.org/abs/2006.03654.

[14] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. arXiv:1909.11942, 2020. Available: https://arxiv.org/abs/1909.11942.

[15] Hugging Face. *ArBert/albert-base-v2-finetuned-ner* Available: Hugging Face [Online; accessed 30-April-2024].

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv:1907.11692, 2019. Available: https://arxiv.org/abs/1907.11692.

Appendix

Algorithm 2 PII Mask under detection tag

Require: Stored PII Entities - a list of detected PII entities, OriginalText - the original text containing PII

```
# Sort entities based on their start positions
FormattedResults ← SortEntitiesByStart(Entities)
# Initialize offset
Offset ← 0
# Iterate through each entity
for all Result in Stored PII Entities do
    # Extract entity tag
    EntityTag ← ExtractEntityTag(Result)
    # Calculate start and end positions considering the offset
    Start ← Result.Start + Offset
    End ← Result.End + Offset
    # Replace the entity with its tag in the original text
    MaskedText ← ReplaceEntityWithTag(OriginalText, EntityTag, Start, End)
    # Update the offset
    Offset ← Offset + (Length(EntityTag) + 2) - (Result.End - Result.Start)
end for
# Return the masked text
MaskedText ← MaskedText
```

Algorithm 3 PII Mask under Redaction

Require: Stored PII Entities - a list of detected PII entities, OriginalText - the original text containing PII

```
# Sort entities based on their start positions
FormattedResults ← SortEntitiesByStart(Entities)
# Initialize offset
Offset ← 0
# Iterate through each entity
for all Result in Stored PII Entities do
    # Extract entity tag
    EntityTag ← ExtractEntityTag(Result)
    # Calculate start and end positions considering the offset
    Start ← Result.Start + Offset
    End ← Result.End + Offset
    # Replace the entity with its tag in the original text
    MaskedText ← ReplaceEntityWithTag(OriginalText, [Redaction], Start, End)
    # Update the offset
    Offset ← Offset + (Length('Redaction') + 2) - (Result.End - Result.Start)
end for
# Return the masked text
MaskedText ← MaskedText
```

Algorithm 4 PII Mask under obfuscation

Require: Stored PII Entities - a list of detected PII entities, OriginalText - the original text containing PII

```
# Sort entities based on their start positions
FormattedResults ← SortEntitiesByStart(Entities)
# Iterate through each entity
for all Result in Stored PII Entities do
    # Extract entity tag
    EntityTag ← ExtractEntityTag(Result)
    # Calculate start and end positions
    Start ← Result.Start
    End ← Result.End
    # Mask the PII word by replacing all characters except the first one with 'x's
    word ← Result['word'][1:]
    MaskedWord ← word[0] + 'x' * (len(word) - 1)
    MaskedText = OriginalText[:start] + MaskedWord + OriginalText[end:]
end for
# Return the masked text
MaskedText ← MaskedText
```