NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Final Project

Personally Identifiable Information (PII) Data Detection

School of Computer Science and Engineering
Programme: MSAI
Date: 21 April 2024
Team Members:

Tan Jie Heng Alfred - G2304193L
Tan Shu Ting Germaine - G2304211B
Wong Qi En Joann - G2304203J
Sui Jinhong - G2302859J
Siew Tze Kang, Julian - G2303643F
Pham Minh Khoi - G2303923E

Abstract

In this project, we explore various methods for Named Entity Recognition (NER) in the context of detecting Personally Identifiable Information (PII) in educational datasets. Our objective is to determine that best performing model during training. We experiment with two Large Language ModelS (LLMs), DeBERTa and T5, and tested them on different configurations, including different loss functions, fine-tuning using an added classifier head, fine-tuning of the entire LLM with the classifier head, and training the model on Low-Rank Adaptation (LoRA). Among the experiments that we ran, we find that fine-tuning the entire T5 model with a classifier head, supported by a LoRA configuration, produces the best training results. Finally, we propose methods to further extend research in the field of PII for the education domain.

1 Introduction

Personally Identifiable Information (PII) are data that can be used to identify a specific person. This includes word entities such as names, addresses, national identification numbers, phone numbers, and email addresses. In educational research, PII is a major roadblock to sharing educational data. This is because PII data needs to be removed from data before being uploaded to educational datasets, but this is traditionally a manual and time-consuming task.

The availability of large educational datasets is important for the analysis and advancement of education research. Being able to effectively remove PII from educational data facilitates valuable research without compromising student safety. Some areas of educational research include learning patterns and effectiveness of teaching methods. Such research would have a long-lasting impact on the quality of education of future generations.

1.1 Limitations of Current Methods for PII Detection

Traditionally, PII is manually removed from datasets. This involves having human agents going through large datasets to identify and remove PII. While it is largely reliable, this process is time-consuming and very labour intensive. In addition, as with any human process, it is subject to human error and sensitive information can be missed. Hence, this manual method is not scalable and is insufficient for large-scale, efficient PII removal.

Another method is the use of predefined rules and signatures to identify PII. Some examples include using regular expressions to look for standardised email formats, phone number lengths, and national ID structures. While effective for some PII, this method is ineffective for less-structured PII, such as names with multiple first, last, and middle names, and addresses that do not follow structure. Given that PII formats can also evolve and change, constant rule adjustments are required. For instance, given that PII is broadly defined to be any information that can be used to identify a person, new PII types can emerge as technology develops over time, such as social media handles and usernames.

1.2 Deep Learning Methods for NER Tasks

Named Entity Recognition (NER) is a sub-task in Information Extraction that classifies named entities into predefined categories. PII detection can be considered an NER task, for which deep and machine learning methods have been increasingly adopted in recent years. However, while deep and machine learning models for NER tasks are much more efficient than manual and predefined methods, they are difficult to train and do not necessarily perform well. Thus, the objective of this project is to explore deep learning methods that can improve the accuracy of NER models for PII.

In unstructured data like documents and essays, it can be challenging for the model to understand context well enough to detect PII. For example, a student's name is considered PII and should be removed, but a known figure's name such as "Donald Trump" is not a privacy concern. In another example, the word "Chloe" could be a student's name which should be removed, or could also refer to the fashion brand that should not be removed.

In the context of PII, one important consideration is managing the trade-off between accuracy and privacy. False positives might flag harmless data as PII and remove it. This affects readers' understanding of the original document and hinders the research that it was meant to help. On the other hand, false negatives will miss real PII, leaving sensitive data vulnerable to be exploited or making an organisation liable to non-compliance.

2 Data

2.1 PII Dataset

Our initial PII dataset was provided by Kaggle and consists of essays written by students enrolled in a massive open online course. All of the essays were written in response to a single assignment prompt, which asked students to apply course material to a real-world problem (Holmes 2024). The initial data consists of 6807 documents/ essays together with the respective tokens tokenized using spaCy, with a binary column indicating the presence of trailing whitespace. The labels provided are on a token level (i.e. the number of labels for each document corresponds to the number of tokens for that particular document) and consists of seven classes of PII (Holmes 2024).

Token labels were presented in the BIO (Beginning, Inner, Outer) format. The PII type was prefixed with "B-" signalling the Beginning of an entity, "I-" signalling the continuation of an entity and "O" signalling that the tokens are not PII (Holmes 2024).

- NAME_STUDENT - The full or partial name of a student that is not necessarily the author of the essay. This excludes instructors, authors, and other person names. *Prefixed with either B- or I-*
- EMAIL - A student's email address. *Prefixed with B- only*

- USERNAME - A student's username on any platform. *Prefixed with B- only*
- ID_NUM - A number or sequence of characters that could be used to identify a student, such as a student ID or a social security number. *Prefixed with either B- or I-*
- PHONE_NUM - A phone number associated with a student. *Prefixed with either B- or I-*
- URL_PERSONAL - A URL that might be used to identify a student. *Prefixed with either B- or I-*
- STREET_ADDRESS - A full or partial street address that is associated with the student, such as their home address. *Prefixed with either B- or I-*

USERNAME and EMAIL do not have an "I-" prefix as they only come in a single token, resulting in a total of 13 tokens as listed above. We also note that the labels in the data provided are highly imbalanced as shown below (in

Labels	Proportion in dataset (%)
O	99.945138
B-STUDENT_NAME	0.027341
I-STUDENT_NAME	0.021953
B-URL_PERSONAL	0.002203
B-ID_NUM	0.001562
B-EMAIL	0.000781
I-STREET_ADDRESS	0.000401
I-PHONE_NUM	0.000300
B-USERNAME	0.000120
B-PHONE_NUM	0.000120
B-STREET_ADDRESS	0.000040
I-URL_PERSONAL	0.000020
I-ID_NUM	0.000020

Table 1: The proportion of each type of label in the dataset.

2.2 Data Preprocessing

The original data provided made use of the spaCy word tokenizer to tokenize individual word tokens from each sentence. However, we recombined the spaCy-tokenized words to use Byte Pair Encoding (BPE) instead. BPE ensures that the most common words are represented in the vocabulary as a single token while rare words are broken down into two or more subword tokens. In addition, different neural networks provide tailored tokenizers which allow the final tokens to better match the embedding layer. To reduce the odds of out-of-vocabulary tokens, we made use of the tokenizers from the respective pre-trained model in our following experiments, all of which are different variations of BPE.

In the original data provided, each word and punctuation was tagged to a label. In BPE, some words were broken down into multiple chunks while some word-punctuation combinations were removed from the documents. This resulted in misalignment between the original labels and the BPE tokens. Additional data processing had to be done to reconcile this difference.

We first combined each token with the trailing white spaces to obtain the initial document before applying the respective tokenizers. The tokenizers provided us with additional outputs such as offset mapping, attention mask and input_id (numeric representation of the respective tokens specific to the respective embedding layer of the pre-trained model). The attention mask and input_id were used as input for the pre-trained model while the offset mapping was used as an index to identify which word and punctuation were split or removed. This allowed us to replicate the same logic on the labels. Next, we applied a one-hot encoding to the labels.

3 Experiments

In this section, we conducted a series of experiments with various LLMs. The main objective of this project is to experiment with model training and hyperparameter tuning to determine the best model for our NER task in the PII domain.

We started with a specified LLM and fine-tuned the model by experimenting with different loss functions, and model configurations. We utilised the Hugging Face Transformers library to load pre-trained models and fine-tune them for our NER task. Across all the experiments, we fixed the following:

- Adam optimiser with learning rate = 0.001 and weight decay = 0.0005
- Cosine annealing learning rate scheduler
- Ran 5 epochs in training (due to GPU and time constraints)

3.1 Experimental Setup 1

For the following set of experiments, we used the DeBERTa-v3-small pre-trained model (He 2021) with $\sim$141 million parameters as our LLM. We used the AutoTokenizer class from the Transformers library and applied the tokenizer from the DeBerta-v3 model. The objective of this set of experiments is to compare how fine-tuned LLMs perform when they are trained on different loss functions.

DeBERTa Fine-Tuned with Cross Entropy Loss

In our first experiment, we fine-tune the pre-trained DeBERTa model for our NER task by adding a classification head to perform label classification at a token level on our PII dataset.

We use AutoModelForTokenClassification, which is a generic model class that loads the pre-trained layers of the DeBERTa-v3-small model, and automatically adds a token classification head on top of it (HuggingFace 2024). This

classification head consists of a linear layer that maps the high-dimensional representations produced by DeBERTa to the 13 label classes specified in our PII NER task. This classification head is specifically tailored for token-level classification tasks (such as NER), where each token in an input sequence is individually classified into one of the 13 predefined labels. The model outputs a score for each category, and these scores are then transformed into probabilities using a softmax layer.

Given that we froze the parameters of the pre-trained DeBERTa model, after adding the classification head, which starts with randomly initialised weights, we have 9,997 trainable parameters for the fine-tuning step. In the fine-tuning step, we train the model on our labelled PII dataset using a cross-entropy loss function to measure error in predictions.

Cross entropy loss is widely used as a loss function for multi-class classification tasks and measures the performance of a classification model, where the output is a probability between 0 and 1. We use the following formula for cross entropy loss:

$$L_{\text{CE}}(p, y) = -\sum_{c=1}^{C} y_c \log(p_c) \tag{1}$$

where

- $L_{\text{CE}(p,y)}$ is the cross entropy loss function,
- p_c is the predicted probability of the observation belonging to class c,
- y_c is the ground truth label for class c, which is 1 if the ground truth class is c and 0 otherwise,
- and the sum is taken over all classes C, from 1 to C.

Cross entropy loss increases as the predicted probability diverges from the ground truth label.

DeBERTa Fine-Tuned with Focal Loss

Next, we replicate the same experimental set-up, but replace the cross entropy loss function with focal loss. This is because while cross entropy loss is generally more widely used, it is more suited to tasks where classes are balanced. On the other hand, focal loss is a modified version of cross entropy loss and addresses class imbalance. It is designed to down-weight the loss contributed by easy examples and focus training on hard examples.

We hypothesise that training our model on focal loss rather than cross entropy loss would produce better results because in our PII dataset, label classes such as "O" are heavily over-represented, while other label classes such as "URL_PERSONAL" are underrepresented. Given our imbalanced dataset, focal loss can prevent the overwhelming number of easy examples from dominating the training of the model. It helps in focusing the model's attention on difficult-to-classify instances, which is beneficial when some classes are rarer than others.

We use the following formula for focal loss:

$$L_{\text{focal}}(p, y) = -\sum_{c=1}^{C} y_c (1 - p_c)^{\gamma} \log(p_c) \tag{2}$$

where

- $L_{\text{focal}}(p, y)$ is the focal loss function,
- p_c is the predicted probability of the observation belonging to class c,
- y_c is the ground truth label for class c, which is 1 if the true class is c and 0 otherwise,
- $(1 - p_c)^{\gamma}$ is the modulating factor to adjust the importance of correcting misclassified examples, with γ being a tunable parameter that controls the focusing strength,
- and the sum is taken over all classes C.

In comparison with cross entropy loss, focal loss adds a modulating factor $(1 - p_c)^{\gamma}$ to the standard cross entropy criterion, which reduces the loss contribution from easy examples and increases the importance of correcting misclassified examples. This adjustment would help in training our model, where there is a significant imbalance among classes, as it prevents the majority classes from overwhelming the minority classes during training.

In our experiment, we used a modulating factor where $\gamma = 2$. This aggressively down-weights the loss contributed by well-classified examples and makes the loss more sensitive to misclassified or difficult examples. Setting $\gamma = 2$ makes the training focus much more on difficult examples and reduces the influence of easy examples in the learning process.

3.2 Experimental Setup 2

T5 Fine-Tuned with Focal Loss

Based on the results in Part 1 of our experiments (discussed in detail in the Results and Discussion sections), we found that focal loss produces much better results with our data. Hence, we decided to use only focal loss in all our subsequent experiments.

Next, we decided to experiment with a smaller LLM. Our motivation for using a smaller LLM is because our PII dataset is for the educational domain. In the context of the education sector, there are generally less resources available

for deployment of large scale, complex models. Hence, ease and speed of computation and deployment are important factors to consider, while also not compromising too much on model performance.

Therefore, we chose to use T5 as it is a lightweight model with relatively fewer parameters. Unlike the DeBERTa-small model with ∼141 million parameters, the T5-small model has ∼36 million parameters (Raffel 2020). The objective of this experiment is to investigate how LLMs of different sizes perform differently in NER tasks.

Like the experiments in Part 1, we froze the parameters of the pre-trained T5 LLM, used a T5 tokenizer, and added a classifier head on top of the T5 model for fine-tuning, which resulted in 6,669 trainable parameters. We trained the model on focal loss.

3.3 Experimental Setup 3

Fine-Tuning Entire T5 Model with LoRA
Moving into our final experiment, we wanted to investigate the difference between fine-tuning only the classifier head and the entire model. Due to the limited computational resources, we decided to base our investigation on the lighter weight T5 model. However, the model still has ∼36 million parameters, making training prohibitively expensive and long. To address this, we adopted Low-Rank Adaptation (LoRA), which is a technique designed to address the high computational and memory costs, among other things, involved in fine-tuning large models (Hu 2022).

Suppose that the pre-trained model has weights W, and a downstream task requires an optimal weights W'. Then, in fine-tuning the model, we hope to learn ΔW, such that

$$W' = W + \Delta W \tag{3}$$

LoRA's approach is to decompose the weight updates, ΔW, into two low-rank matrices, A and B, while freezing the pre-trained weights W. This allows us to update only the low-rank matrices, making training more computationally efficient than updating the higher dimensional ΔW directly. Furthermore, the decrease in the number of trainable parameters (i.e. from elements in ΔW to elements in A and B) reduces the memory footprint when training the model. Besides, the lower number of trainable parameters also addresses potential over-fitting issues, which might be pertinent in our relatively small dataset.

Following the authors in (Hu 2022), we apply LoRA to the attention block in T5. This corresponds to the modules {`'q', 'k', 'v', 'o'`}. We did not apply LoRA to the smaller classifier head, as it might impair the performance of the model on the downstream task. Instead, we unfroze the otherwise frozen classifier layer for further fine-tuning. We hypothesised that this would improve the runtime of the model as well as its performance.

4 Results

In this section, we report the results of our experiments. The results are reported in the following tables to allow for easy comparison within each set of experiments.

The first column, "Model", indicates the LLM used in the experiment. The "Loss" column indicates the loss function the model was trained on. The column "Full Model or Classifier" indicates whether we fine-tuned only the classifier head or fine-tuned the entire model. The column "Num Trainable Parameters" tells us the number of trainable parameters during training, and the "Val F5" column reports the F5 metric score during the validation phase of training.

Model	Loss	Full Model or Classifier	Num Trainable Parameters	Val F5
DeBERTa	CE	Classifier	9,997	0.737
DeBERTa	Focal	Classifier	9,997	0.902

Table 2: Effect of loss functions on model performance

Model	Loss	Full model or Classifier	Num trainable Parameters	Val F5
DeBERTa	Focal	Classifier	9,997	0.902
T5	Focal	Classifier	6,669	0.733

Table 3: Experiment results of DeBERTa vs T5 model architectures

Model	Loss	LORA	Full model or Classifier	Num trainable Parameters	Val F5
T5	Focal	Yes	Full model	1,579,533	0.909
T5	Focal	No	Classifier	6,669	0.733

Table 4: Model performance when training classifier head only vs training entire model with Lora

5 Discussion

5.1 Cross Entropy vs Focal Loss Functions

From the results in Table 2, we observe that focal loss does provide significant improvement to our DeBERTa model performance. We attribute this improvement to the suitability of focal loss for datasets with imbalanced class labels, such as our PII dataset. The F5 score of the model trained on focal loss is 0.902, an improvement over the model trained on cross entropy loss. This suggests that the modulation factor in the focal loss has been particularly effective in training our imbalanced dataset, helping the model develop a more refined understanding of difficult and underrepresented label classes. This has developed a more robust model that generalises better across the entire dataset, particularly over minority classes.

Given the superior performance of the model trained on focal loss, we carry out subsequent experiments using only focal loss.

5.2 DeBERTa Fine-Tuned vs T5 Fine-Tuned

We now compare the results of two LLMs, DeBERTa and T5, that have both been trained on focal loss and had their

classifier heads fine-tuned. The results in Table 3 show that the T5 model produces a much lower model performance with an F5 score of 0.733, compared to the DeBERTa model with an F5 score of 0.902.

This could be because T5 is a much smaller model with much fewer parameters than DeBERTa. The larger number of parameters in DeBERTa likely allows it to model more complex relationships and patterns within the data. This increased complexity might also help it develop a more nuanced understanding of textual features, which is necessary for carrying out the NER task. Conversely, T5's smaller parameter size, while providing the advantage of being more computationally lightweight, could have limited its capacity in capturing more intricate depths of the NER task as well as the full range of linguistic features. This could have contributed to the lower F5 score. The results of this experiment suggest that the parameter size of an LLM can have a significant effect on its performance, particularly in tasks that require a high degree of nuanced text understanding.

We also consider secondary reasons relating to the architecture of the LLMs. For instance, DeBERTa leverages a disentangled attention mechanism, which disentangles each word into its content and relative position vectors, unlike the usual attention mechanism. The structural advantage allows DeBERTa more freedom to capture more nuanced relationships and complex dependencies between tokens, and subtle linguistic cues within the text. On the other hand, T5, being a text-to-text framework, may not specialise as efficiently in token-level classification tasks in NER, where detailed and contextual understanding of each token's role is essential.

5.3 Fine-Tuning Entire T5 Model with LoRA vs Fine-Tuning Classifier Head Only

Finally, we compare the performance of the T5 model that has been entirely fine-tuned using LoRA versus the T5 model that only has the added classifier head fine-tuned. As previously explained, we chose to use T5 as our LLM in this experiment because of resource constraints, as T5 is a more lightweight LLM.

The results in Table 4 show that fine-tuning the entire T5 model, supported by the LoRA configuration, produces better results, with an F5 score of 0.909. By applying LoRA, we reduced the total number of trainable parameters from $\sim$36 million to $\sim$1.5 million. We obtained poorer results from fine-tuning only the classifier head (while keeping the T5 model's pre-trained parameters frozen), which gave an F5 score of 0.733.

The results of this experiment suggest that fine-tuning the entire model serves to improve model performance. Restricting the fine-tuning to only the classifier head evidently limits the model's capacity to fully learn from the dataset. Thus, the flexibility to also efficiently adjust the components of the feature extractor enables the model to develop a more refined understanding of the data, leading to better classification of labels in terms of precision and recall. Finally, it also demonstrates that the adoption of LoRA still preserves model performance while significantly reducing the number of trainable parameters, and by extension, computational costs.

6 Conclusion

Out of all our experiments, we achieved the best F5 score of 0.909 by fine-tuning the entire T5 model with a classifier head, supported by LoRA. This result demonstrates the suitability of using focal loss as our loss function, the effectiveness of fine-tuning the entire T5 model, as well as the benefit of adopting LoRA in our training.

Finally, it is important to note that in PII for education, there is generally less funding available for research and deployment of the models we have discussed, compared to domains like biomedicine and finance. Hence, computational and deployment costs are important factors to consider, and while the model's ability to perform well and safeguard students' personal information is critical, there needs to be a careful balance struck between model performance and the cost of the model.

7 Future Work

7.1 Extending Our Experiments

Given more time and resources, we would also have liked to re-train the entire DeBERTa model with the classifier head, supported by LoRA. This is because in our first experiment, training only the classifier head for the DeBERTa model already produced rather promising results, with a relatively good F5 score of 0.902. We hypothesise that re-training the entire model would lead to even better results than re-training the entire T5 model, since DeBERTA is a much larger LLM.

7.2 Zero-Shot NER with LLMs

In the last two years, research has proven zero-shot learning with LLMs, particularly with ChatGPT, to be an increasingly viable and robust low resource method for Information Extraction tasks including NER (Li et al. 2023)(Han et al. 2023)(Wei et al. 2023). Our team briefly tested a few PII NER examples by feeding a contextualised prompt to ChatGPT-4 and obtained promising preliminary results (Appendix A).

Given the favourable results produced by many recent papers with ChatGPT, zero-shot NER should be further explored and is especially relevant for PII considering the current deficiency of labelled data in many PII domains. However, one barrier we experienced in conducting proper experiments using zero-shot NER with ChatGPT was the high monetary costs associated with calling the ChatGPT API. Given sufficient resources and additional research into prompt engineering, we hypothesise that zero-shot NER with ChatGPT could potentially perform exceedingly well with very low time and computational costs.

7.3 Ensemble Methods

Ensemble methods such as voting and stacking have shown to be able to improve model performance. Voting involves training multiple models (e.g. BART, RoBERTa, LSTM) and combining their predictions using majority voting or more sophisticated weighting schemes. This helps average out individual model errors and can lead to more robust PII identification. Stacking involves training a meta-learner (which could be a simpler model) that takes the predictions of different models as input and learns how to best combine them. This allows the meta-learner to identify patterns in how different models succeed or fail in specific cases.

It should be noted that while ensemble methods do produce better performance, they can be computationally expensive and time consuming to train.

8 Group Contributions

Members	T5	DeBERTa	Model and data loading	LoRA	Write Report	Presentation	Creating ppt slides
Alfred	X	X	X	X	X	-	-
Jinhong	X	X	X	X	X	-	-
Joann	X	-	-	X	X	-	-
Julian	-	-	-	-	-	-	X
Germaine	-	-	-	-	X	X	X
Khoi	-	-	X	-	X	-	-

References

Han, R.; Peng, T.; Yang, C.; Wang, B.; Liu, L.; and Wan, X. 2023. Is Information Extraction Solved by ChatGPT? An Analysis of Performance, Evaluation Criteria, Robustness and Errors. arXiv:2305.14450.

He, P. 2021. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *ICLR 2021*.

Holmes, L. 2024. *The Learning Agency Lab - PII Data Detection.*

Hu, E. J. 2022. LoRA: Low-Rank Adaptation of Large Language Models. *ICLR 2022*.

HuggingFace. 2024. AutoModel for Token Classification. Hugging Face's Transformers Documentation. Available online: https://huggingface.co/transformers/v3.0.2/model_doc/auto.html#automodelfortokenclassification [Accessed: 22-April-2024].

Li, B.; Fang, G.; Yang, Y.; Wang, Q.; Ye, W.; Zhao, W.; and Zhang, S. 2023. Evaluating ChatGPT's Information Extraction Capabilities: An Assessment of Performance, Explainability, Calibration, and Faithfulness. arXiv:2304.11633.

Raffel, C. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research, Volume 21*.

Wei, X.; Cui, X.; Cheng, N.; Wang, X.; Zhang, X.; Huang, S.; Xie, P.; Xu, J.; Chen, Y.; Zhang, M.; Jiang, Y.; and Han, W. 2023. Zero-Shot Information Extraction via Chatting with ChatGPT. arXiv:2302.10205.

A Zero-Shot NER with ChatGPT-4

Contextualisation = " " "

The following text contains personally identifiable information (PII) of students.
Assign labels to the following seven types of PII:
NAME_STUDENT - The full or partial name of a student that is not necessarily the author of the essay. This excludes instructors, authors, and other person names.
EMAIL - A student's email address, usually containing the characters '@' and '.com'.
USERNAME - A student's username on any platform.
ID_NUM - A number or sequence of characters that could be used to identify a student, such as a student ID or a social security number.
PHONE_NUM - A phone number associated with a student.
URL_PERSONAL - A URL that might be used to identify a student.
STREET_ADDRESS - A full or partial street address that is associated with the student, such as their home address.

Token labels are presented in BIO (Beginning, Inner, Outer) format. The PII type is prefixed with 'B-' when it is the beginning of an entity. If the token is a continuation of an entity, it is prefixed with 'I-'. Tokens that are not PII are labelled 'O'.

" " "

Text = " " "

Diego Estrada Design Thinking Assignment Visualization Tool Challenge & Selection

" " "

Prompt = Contextualisation + Text

Output:

- Diego - B-NAME_STUDENT
- Estrada - I-NAME_STUDENT
- Design - O
- Thinking - O
- Assignment - O
- Visualization - O
- Tool - O
- Challenge - O

- & - O
- Selection - O

Using zero-shot prompting for NER, ChatGPT-4 correctly predicted the labels for all the given words in the above example.