1. Abstract

Link to Video: https://mediaspace.illinois.edu/media/t/1_smoraeaf
Link to GitHub Repo: https://github.com/Samyak005/deidentification-medical-records

2. Introduction

Medical reports often contain *personally identifiable information (PII)* (like names, addresses, phone numbers) and *personal health information (PHI)* (like medical history, diagnoses). It is important to maintain confidentiality to protect patients' trust. There are several laws like **HIPAA** (in the US), **GDPR** (in Europe), that enumerate very strict rules for personal health data. A patient's information if exposed can lead to identity theft, discrimination, or stigma (e.g., around mental health, HIV status).

However, large-scale analysis of health problems to establish trends and identify patterns in a society requires a deeper analysis of medical records. Therefore, anonymization of health records is an important problem to solve.

Citation: Transformer-DeID: Deidentification of free-text clinical notes with transformers Published: Nov. 2, 2023. Version: 1.0.0 (https://physionet.org/content/transformer-deid/1.0.0)

Creation of Synthetic Dataset

Despite best efforts, the dataset associated with the paper could not be sourced. In absence of the original data set, synthetic data with 1000 records was created using the sample available with the paper. The scope of the work was to demonstrate the efficacy of the pre-trained models provided by the authors of the original paper on the new synthetic dataset.

3. Methodology

3.a Environment - Python version `3.11.12`, in the default environment, Google Colab

Dependencies/packages needed- PyTorch, transformers, datasets, seqeval, scikit-learn, evaluate

3.b Data

Path in repository: https://github.com/Samyak005/deidentification-medical-records/tree/main/dataset

The synthetic dataset consists of 2 JSON files - annotations.json and discharge_summaries.json
The discharge summaries file consists of a document_id and the text of the medical record.
The annotations file contains the annotation_id, start_index, stop_index, entity, entity_type, and the document_id.
Dataset Statistics:

Metric	Value
Number of Documents	1000
Number of Annotations	9000
Average Annotations per Document	9.0
Average Document Length (words)	205.9

Entity Type	Count
NAME	2000
AGE	1000
DATE	3000
IDNUM	1000
LOCATION	1000

PHONE	1000

"text": "Name: Ashley Wolfe   Unit No: 1110277\nAdmission Date: 23/08/2023   Discharge Date: 25/08/2023\nDate of Birth: 13/07/2009   Age: 57   Sex: F\nService: Paediatrics\nAttending: Paula Sutton\n\nChief Complaint: Patient presented with complaints relevant to paediatrics evaluation.\n\nHistory of Present Illness:\nAshley Wolfe, a 57-year-old logistics and distribution manager from Robertburgh, KS, was admitted with a several-day history of symptoms requiring paediatrics evaluation.\nSymptoms were progressive and included complex features related to the underlying condition. On evaluation, the patient reported symptom details including variations, associated features, and previous management attempts.\n\nPhysical Examination and Diagnostic Findings:\nThorough physical evaluation was conducted, revealing relevant signs supportive of the working diagnosis.\nLaboratory studies, imaging (e.g., CT/MRI/Ultrasound/X-ray), and specialty consultations were pursued to refine diagnosis.\n\nDiagnosis:\nBased on history and investigations, the final diagnosis was determined as a condition commonly managed in paediatrics with consideration of differential diagnoses.\n\nTreatment and Hospital Course:\nThe patient underwent appropriate pharmacologic and/or procedural interventions. Response to treatment was closely monitored.\nPain control, supportive therapy, and targeted interventions were used. The patient remained hemodynamically stable throughout.\n\nDischarge Plan and Follow-Up:\nPatient was discharged in stable condition. Advised to continue medications and follow up at the outpatient paediatrics clinic.\nFollow-up appointment scheduled at: Robertburgh, KS clinic. Contact: 053-606-5681x5409."

{   "annotation_id": "T1",   "start": 6,   "stop": 18,   "entity": "Ashley Wolfe",   "entity_type": "NAME",   "document_id": "ex_ds_1" },
{   "annotation_id": "T2",   "start": 130,   "stop": 132,   "entity": "57",   "entity_type": "AGE",   "document_id": "ex_ds_1" }

3.c Model

Original Repo Link: https://github.com/kind-lab/transformer-deid

The authors used 3 transformer models - BERT, RoBERTa, and DistilBERT.
BERT (Bidirectional encoder representations from transformers) has been trained using masked language modeling and next sentence prediction on the BookCorpus and English Wikipedia texts

DistilBERT is a distilled version of BERT which is trained on the same corpi as BERT, but is pre-trained with distillation loss and cosine embedding loss.

RoBERTa is the robustly optimized BERT pre-training approach which made some improvements on BERT, including dynamic masking, modifying the input format, and utilizing large-batch training.

Inputs:

Text data: Discharge summaries containing PHI

Annotations:

1. PHI type (e.g., "name", "date")
2. Start/end character indices
3. Text snippet of PHI

Outputs:

1. Token-level predictions for PHI classes
2. Binary evaluation (PHI vs non-PHI) despite multi-class labeling

Techniques used:

Entity Label Mapping:

Original PHI categories mapped to 7 classes: ['name', 'date', 'age', 'location', 'ID', 'contact', 'profession'] and "O" label for non-PHI tokens

Tokenization: BERT and DistilBERT use WordPiece tokenization, while RoBERTa uses Byte-Pair Encoding (BPE)

The Hugging Face models available as part of KindLab did not provide a tokenizer file. Hence, the tokenizer of the base classifier, namely, BERT, RoBERTa, and DistilBERT, respectively, were used along with their pre-trained models for testing and finetuning.

tokenizer = *AutoTokenizer.from_pretrained("bert-base-cased")*.

To use the tokenizer of the base classifier, alignment of labels is required. We tokenized text and aligned character-based entity labels with the corresponding tokens produced by BERT tokenizer. For token-level classification tasks like Named Entity Recognition (NER), each token needs a corresponding label. Initially, every token is labeled as "O" (which means "Outside" in BIO tagging—no entity).

Pretrained models were available from the original paper [1]. Three models were available: BERT [2], RoBERTa [3], and DistilBERT [4]. All three models were used for prediction and fine-tuning with the synthetic data.

Fine Tuning Pre-trained Models and Fresh training:
We explored two approaches:

a. Fine tune the three pre-trained models available along with original paper, with the synthetic data and measure the performance.
b. Train the three base models BERT, RoBERTa, DistilBERT with the synthetic data and measure the performance.

3.d. Training

The train, validation, test split of 80%, 10%, 10% respectively and test data evaluation on 100 test points, were kept fixed.

3.d.1. Hyperparameter Tuning: Fine Tuning of BERT-based LLM with Synthetic Data

We experimented with several variations of the LLM training hyperparameters during the finetuning of the pre-trained model KindLab/bert-deid. In the hyperparameter tuning, we explored the impact of varying learning rates, batch sizes, and training epochs on evaluation metrics such as precision, recall, F1-score, and accuracy. Also, we explored cosine and linear dynamic learning rates with warm up = 0.3. Weight decay regularization was included in an experiment to explore further improvements.

Exp ID	Learning Rate	Batch Size	Epochs	Eval Accuracy	Eval Precision	Eval Recall	Eval F1
1	2e-5	**8**	3	**0.9997**	**0.9853**	**0.9977**	**0.9915**
2	2e-5	16	3	0.9996	0.9809	0.9977	0.9892
3	2e-5	32	3	0.9995	0.9721	0.9943	0.9831
2	2e-5	16	3	0.9996	0.9809	0.9977	0.9892
4	2e-5	16	**5**	0.9996	**0.9853**	0.9965	**0.9909**

5	2e-5	16	4	0.9996	0.9820	0.9988	0.9904
6	2e-5	8	5	0.9997	**0.9853**	0.9977	**0.9915**
7	1e-5	8	5	0.9996	0.9831	0.9977	0.9903
8	1e-6	8	5	0.9983	0.9298	0.9817	0.9551

The learning rate of 2e-5 produced a high F1 score above 0.9915, which was higher than 1e-5 and 2e-6. Batch size 8 achieved the best F1 scores (over 0.9915), outperforming larger batch sizes of 16 and 32. Training for 5 epochs outperformed 3 or 4 epochs significantly. Models trained for 5 epochs recorded F1 scores of 0.9909 and higher precision values, demonstrating more complete training without overfitting.
Best Performing Configuration - Learning Rate: 2e-5, Batch Size: 8, Epochs: 5

The conclusion from the first stage of experiments was to train for longer (7, 10 epochs) and introduce dynamic learning rate schedulers.

Exp ID	Learning Rate		Epochs	Eval Accuracy	Eval Precision	Eval Recall	Eval F1
7	1e-5	fixed	5	0.9996	0.9831	0.9977	0.9903
10	1e-5	linear	5	0.9996	0.9820	0.9977	0.9898
11	1e-5	linear	7	0.9996	0.9842	0.9965	0.9903
12	1e-5	linear	10	0.9996	0.9820	0.9988	0.9904
13	**2e-5**	**linear**	**5**	**0.9997**	**0.9853**	**0.9977**	**0.9915**
14	2e-5	linear	7	0.9996	0.9853	0.9965	0.9909
15	2e-5	linear	10	0.9996	0.9831	0.9988	0.9909
6	**2e-5**	**fixed**	**5**	**0.9997**	**0.9853**	**0.9977**	**0.9915**
16	2e-5	cosine	5	0.9996	0.9853	0.9965	0.9909
17	2e-5	Fixed with weight decay 0.1	**5**	**0.9997**	**0.9853**	**0.9977**	**0.9915**

Note: Batch size was 8 for all the experiments above

Experiments with the learning rate show that training with 5 epochs resulted in better F1 score (varying learning rates and dynamic learning rates did not help). 2e-5 learning rate dominated learning rate of 1e-5. With learning rate of 2e-5 and Epochs =5, training with linear learning rate matched the fixed learning rate performance.
Best Performing Configuration - Learning Rate: 2e-5, Batch Size: 8, Epochs: 5, weight_decay:0.01 (with fixed learning rate, fixed learning rate is selected as it is relatively faster to execute than the linear learning rate even though the F1 score for both is the same)

3.d.2. Experiments and their Computation Time

Initial training was done on CPU (Google Colab 12.7GB RAM), however, compute time for 20% of training data and with just 3 epochs for batch size of 8 was more than 55 minutes, that is 44.45sec per iteration. Hence, we switched to Google Colab TPU T4 environment with 15GB RAM. For all the experiments, we noted

(a) the total compute time of training and evaluation, and

(b) time for all training iterations. As is apparent from the table below, compute time per training iteration was less than 1 sec for batch size of 8.

Exp ID	Learning Rate		Batch Size	Epochs	Computation time (sec)	Compute per training iteration (sec)
1	2e-5	fixed	**8**	3	288.3	247/300 =0.823

2	2e-5	fixed	16	3	249.68	243/150 =1.620
3	2e-5	fixed	32	3	245.63	237/75 = 3.160
2	2e-5	fixed	16	3	249.68	243/150 =1.620
4	2e-5	fixed	16	**5**	400.13	394/250 =1.576
5	2e-5	fixed	16	4	319.00	312/200 =1.56
6	2e-5	fixed	8	5	404.45	398/500 =0.796
7	1e-5	fixed	8	5	410.04	404/500 = 0.808
8	1e-6	fixed	8	5	408.77	403/500 =0.806
10	1e-5	linear	8	5	498.10	469/500= 0.938
11	1e-5	linear	8	7	577.81	572/700= 0.817
12	1e-5	linear	8	10	818.18	813/1000= 0.813
13	**2e-5**	**linear**	8	**5**	**450.93**	**425/500 =0.850**
14	2e-5	linear	8	7	601.73	595/700 =0.850
15	2e-5	linear	8	10	842.43	837/1000 = 0.837
16	2e-5	cosine	8	5	440.89	412/500 = 0.824
17	2e-5	weight decay 0.1	8	**5**	414.45	408/500 =0.816

Computation time is not correlated with learning rate (fixed) in experiments 6,7 and 8. Increase in batch size resulted in small gains in training compute time – 237 sec for batch size of 32 vs. 247sec for batch size of 8. Significant increase in compute time is observed with increase in Epochs, average time per Epoch declined from

(a) 82.3sec in Exp Id 1 to 79.6sec in Exp Id 6,

(b) 93.8 sec in Exp id 10 to 81.3sec in Exp id 12.

Compute time with "linear" change in learning rate is sensitive to the learning rate value (Exp 10, 11, 12 for 1e-5 vs Exp 13,14,15 for 2e-5). Compute time for "cosine" change in learning rate was very similar to that of "linear" change in learning rate (Exp Id 15 and 16). We observe that the impact of weight decay regularization on computation time is minimal (exp Id 6 and 17).

3.d.3. Training Details

LLM training set up including defining the parameters for DataCollator, Training Arguments, provide training and test datasets and call the Train function.

```
# Step 11: Setup Data Collator
data_collator =
DataCollatorForTokenClassification(
    tokenizer,
    padding=True,
    max_length=512,
    return_tensors="pt"
)

# Step 12: Define training arguments
training_args = TrainingArguments(

output_dir="./kindlab_bert_deid_finetuned
",
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    logging_dir="./logs",
    logging_steps=10,
    save_steps=200,
    save_total_limit=2,
    remove_unused_columns=False
)

# Step 13: Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset["train"],
    eval_dataset=dataset["test"],
    data_collator=data_collator,
    compute_metrics=compute_metrics
)
# Step 14: Train
trainer.train()
```

Huggingface's Trainer passes the batch (with input_ids, attention_mask, and labels) and the Hugging Face's AutoModelForTokenClassification calculates the default Loss function CrossEntropyLoss internally. CrossEntropyLoss is perfect for multi-class classification (8 labels) and is considered the best loss function for NER tasks.

3.e Evaluation and Selecting the Metric

All the models were evaluated on four metrics such as precision, recall, F1-score, and accuracy using seqeval *(from seqeval.metrics import classification_report, accuracy_score, f1_score, precision_score, recall_score).*

This is also an imbalanced class problem. Since most tokens are usually O (no entity), and relatively few are actual entities (like NAME, AGE, DATE, etc.), accuracy is misleading. A highly accurate model can just predict O for everything. Failure to recognize meaningful entities is more important.

The core problem in the labelling of the private information in discharge summaries is Named Entity Recognition (NER). We are classifying *tokens* into fine-grained categories (e.g., NAME, AGE, DATE, etc.). Both precision and recall are very important as it is important to ensure correctness of the entity predicted as containing private information as well as how many of the true entities present, did the model successfully predict?

F1 score is the harmonic mean of precision and recall, F1= 2 * (Precision * Recall)/ (Precision + Recall). If the model is very precise but misses a lot of entities (low recall), F1 will drop. If the model finds all entities but also makes a lot of wrong predictions (low precision), F1 will drop. Therefore, high F1 guarantees that both precision and recall are strong. F1 is also the standard metric used in sequence labelling tasks (like NER).

```
model.to(device)
model.eval()
with torch.no_grad():
    for batch in tqdm(test_dataloader,
desc="Evaluating"):
        input_ids =
batch["input_ids"].to(device)
        attention_mask =
batch["attention_mask"].to(device)
        labels =
batch["labels"].to(device)

        outputs =
model(input_ids=input_ids,
attention_mask=attention_mask)
        logits = outputs.logits
        predictions =
logits.argmax(dim=-1)


all_preds.append(predictions.cpu().numpy(
))

all_labels.append(labels.cpu().numpy())
```

4. Results

Synthetic data was evaluated through

(a) the three pre-trained models,

(b) the three pre-trained models after fine-tuning with the training dataset, and

(c) a fresh training of base models (BERT, RoBERTa, DistilBERT) with the training dataset.

The F1 score for evaluation with just the off-the-shelf pre-trained model was in the range of 0.64 to 0.658 which is significantly lower than the F1 score achieved with the same models after fine-tuning with the training dataset. This appears logical because the pre-trained models have not seen the synthetic data and off the shelf scores are low.

Model and Traintype	Eval Accuracy	Eval Precision	Eval Recall	Eval F1
BERT Pretrained	0.9664	0.5747	0.7619	0.6552
RoBERTa Pretrained	0.9671	0.5512	0.7653	0.6409
DistilBERT Pretrained	0.9665	0.5824	0.7562	0.6580
BERT Fine Tuned*	0.9997	0.9853	0.9977	0.9915
RoBERTa Fine Tuned	0.9996	0.9831	0.9988	0.9909
DistilBERT Fine Tuned	0.9996	0.9831	0.9977	0.9903
BERT Fresh Training*	0.9991	0.9892	0.9964	0.9928
RoBERTa Fresh Training	0.9991	0.9873	0.9959	0.9916
DistilBERT Fresh Training	0.9992	0.9895	0.9964	0.9929

* - For BERT, the results reported are for the best-performing set of hyperparameters

Our initial hypothesis was that the pre-trained models when fine tuned with training data should do better than fresh training with the training data. However, the F1 score with fresh training without using the pre-trained models was higher than pre-trained models after fine-tuning. DistilBERT fresh training gave the best F1 score of 0.9929. The class-wise precision, recall, and F1-score are given below-

LABEL	Precision	Recall	F1-score	support
AGE	0.84	0.99	0.91	78
CONTACT	1.00	1.00	1.00	1128
DATE	0.99	0.99	0.99	1696
ID	0.98	0.99	0.98	419
LOCATION	1.00	1.00	1.00	476
NAME	1.00	1.00	1.00	468

In addition to computing these metrics, we also generated the output through the NER pipeline for manual inspection of labels and predicted output.

Ground Truth Annotations	Model Predictions
`NAME: 'Ashley Wolfe' (start: 6, end: 18)`	`NAME: 'Ashley Wolfe' (start: 6, end: 18)`
`AGE: '57' (start: 130, end: 132)`	`ID: '1110277' (start: 31, end: 38)`
`DATE: '23/08/2023' (start: 55, end: 65)`	`DATE: '23 / 08 / 2023' (start: 55, end: 65)`
`DATE: '25/08/2023' (start: 85, end: 95)`	`DATE: '25 / 08 / 2023' (start: 85, end: 95)`
`DATE: '13/07/2009' (start: 111, end: 121)`	`DATE: '13 / 07 / 2009' (start: 111, end: 121)`
`ID: '1110277' (start: 31, end: 38)`	`AGE: '57' (start: 130, end: 132)`
`LOCATION: 'Robertburgh, KS' (start: 373, end: 388)`	`NAME: 'Paula Sutton' (start: 175, end: 187)`
`CONTACT: '053-606-5681x5409' (start: 1665, end: 1682)`	`NAME: 'Ashley Wolfe' (start: 305, end: 317)`
`NAME: 'Paula Sutton' (start: 175, end: 187)`	`AGE: '57' (start: 321, end: 323)`
	`PROFESSION: 'logistics' (start: 333, end: 342)`
	`PROFESSION: 'distribution manager' (start: 347, end: 367)`
	`LOCATION: 'Robertburgh' (start: 373, end: 384)`
	`LOCATION: 'KS' (start: 386, end: 388)`

As an extension of the project, we explored new LLMs, DeBERTa and Electra discriminative. And trained these 2 models with the dataset to determine if they will improve the F1-score.

Model and Traintype	Eval Accuracy	Eval Precision	Eval Recall	Eval F1
DeBERTa fresh training	0.9992	0.9877	0.9957	0.9917
Electra fresh training	0.9992	0.9888	0.9965	0.9927

While in literature, both DeBERTa and Electra discriminative outperform BERT for NER tasks, performance of Electra discriminative on the synthetic dataset was very close to BERT, but F1 score was still lower than that of BERT and DistilBERT. DeBERTa did not perform as well as Electra and BERT.

5. Discussion

Reproducing the paper and its outcome was not possible because the original dataset was not available.

However, generating the synthetic data helped us test the effectiveness of the pre-trained models on the new dataset. Evaluating the new dataset on the pre-trained KindLab models was moderately difficult. While the Hugging Face implementation was available, tokenizers were not available. Hence, the tokenizers of the standard base models, BERT, RoBERTa, and DistilBERT, were used. After initially struggling to resolve the challenge, we managed to implement the alignment of labels. This helped us to complete the evaluation on the synthetic data.

The availability of Google Colab T4 TPU environment helped complete the experiments in time. CPU train time was manifold higher, and it would not have been possible with just the CPU environment.

To reproduce the results of the paper, the authors should enable wider availability of the original dataset.

6. Author Contributions

Samyak - Model Training, Hyperparameter Tuning, Report Writing

Varshini - PyHealth Contribution and Documentation, Video Presentation

7. References

1. https://huggingface.co/KindLab
2. https://huggingface.co/KindLab/bert-deid
3. https://huggingface.co/KindLab/roberta-deid
4. https://huggingface.co/KindLab/distilbert-deid