PRESERVING PRIVACY FOR MEDICAL DATA

Presenter: Hemant Koti

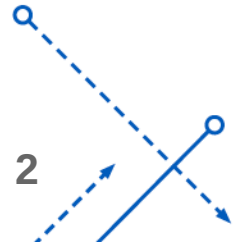
Professor: Sargur N Srihari

Teaching Assistant: Mohammad Abuzar Shaikh

University at Buffalo The State University of New York

Agenda

- Abstract and Introduction
- Goal
- Dataset
- Existing work and limitations
- Proposed Approach
  - PySyft and SyferText
- Code
  - Classifier and Training
- Results
  - Accuracy and Loss
- Conclusion



Abstract and Introduction

- Deep Learning in healthcare can be used to streamline several administrative tasks as well as enabling physicians to make smart decisions based on the data.
- However, healthcare data is highly regulated and private making it inaccessible to perform deep learning tasks.
- In a medical data scenario, the Deep Learning model should be trained on the data without looking at it.
- We aim to explore ways (using PySyft and SyferText libraries) to enable secure and private Deep Learning to decouple sensitive data from the process of model training.

References: https://github.com/OpenMined/pysyft, https://github.com/OpenMined/SyferText

Goal

- Our goal is to build a classifier that correctly classifies the medical specialty based on the transcription text without looking at the dataset itself.
- This also includes the goal to encrypt the model parameters so that reverse engineering cannot be done to decipher dataset inputs.

Dataset

- The original dataset in a raw format is taken from mtsamples.com.
- This dataset contains 40 different medical specialties which in turn contain 5000 transcribed medical reports under all the specialties.
- The raw data has been pre-processed and converted into a CSV format for research purposes on Kaggle.

References: https://www.kaggle.com/tboyle10/medicaltranscriptions#mtsamples.csv

Dataset – EDA

References: https://www.kaggle.com/tboyle10/medicaltranscriptions#mtsamples.csv

Dataset – NLP pre-processing tasks

- We use the files to stop words provided by the clinical concepts repository specifically designed for large medical corpora.
- We generated the vocabulary words based on the classes in Systematized Nomenclature of Medicine (SNMI) data.
- Both the stop words and the vocab words files will be added to the SyferText NLP pipeline (introduced later).

References: https://github.com/kavgan/clinical-concepts, https://bioportal.bioontology.org/ontologies/SNMI

Existing Work

- Existing work on this dataset involves using GloVe and 1D CNN to classify the medical specialty. The benchmark results on this dataset are as follows.

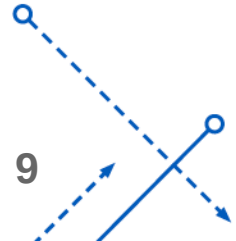
Training Accuracy	Validation Accuracy	Recall	Precision	F1
0.87	0.89	0.88	0.98	0.93

- **Limitations**: However, the model is trained on the private and sensitive information of users. While this is a dataset used for academic research, in real-world scenarios we cannot get any such private and sensitive information.

References: https://nlp.stanford.edu/projects/glove/, https://doi.org/10.36227/techrxiv.12059019.v1

Proposed Approach

- The following are the two **important** steps required to solve our use case.
  - Using the **PySyft** framework create a bigger dataset out of all the client's smaller datasets.
  - Using **SyferText** to prepare and preprocess the text data on the client's machines without revealing it, and without moving any datasets to your machine.
- We simulate an environment where each client owns a part of the full dataset and prepare each worker to perform encrypted training on these datasets.

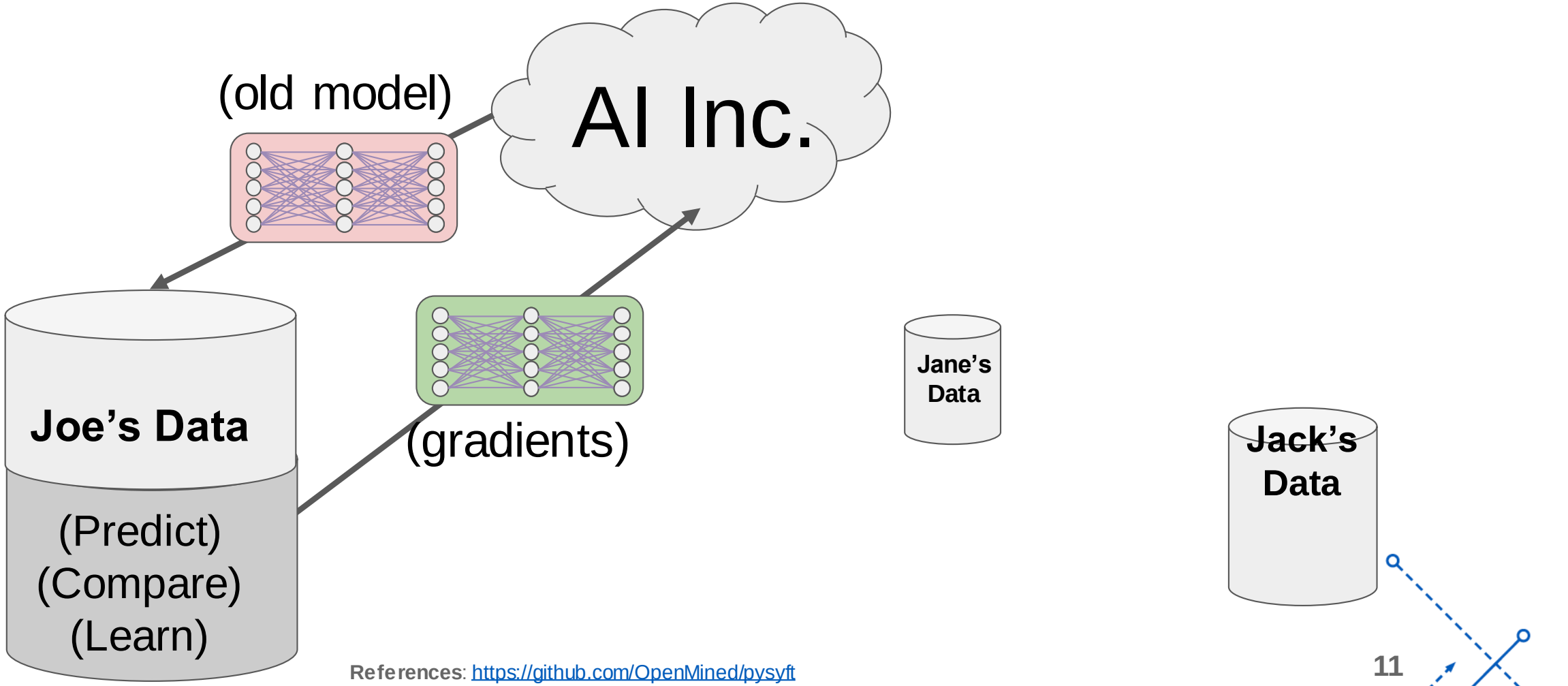


Proposed Approach – PySyft

- PySyft is a Python library for secure and private Deep Learning.
- PySyft decouples private data from model training, using Federated Learning, Differential Privacy, and Encrypted Computation (Multi-Party Computation (MPC)
  - **Federated Learning**: A type of remote execution wherein models are sent to remote data-holding machines for local training. This eliminates the need to store sensitive training data on a central server.
  - **Multi-party computation**: When a model has multiple owners, multi-party computation allows for individuals to share control of a model without seeing its contents such that no sole owner can use or train it.

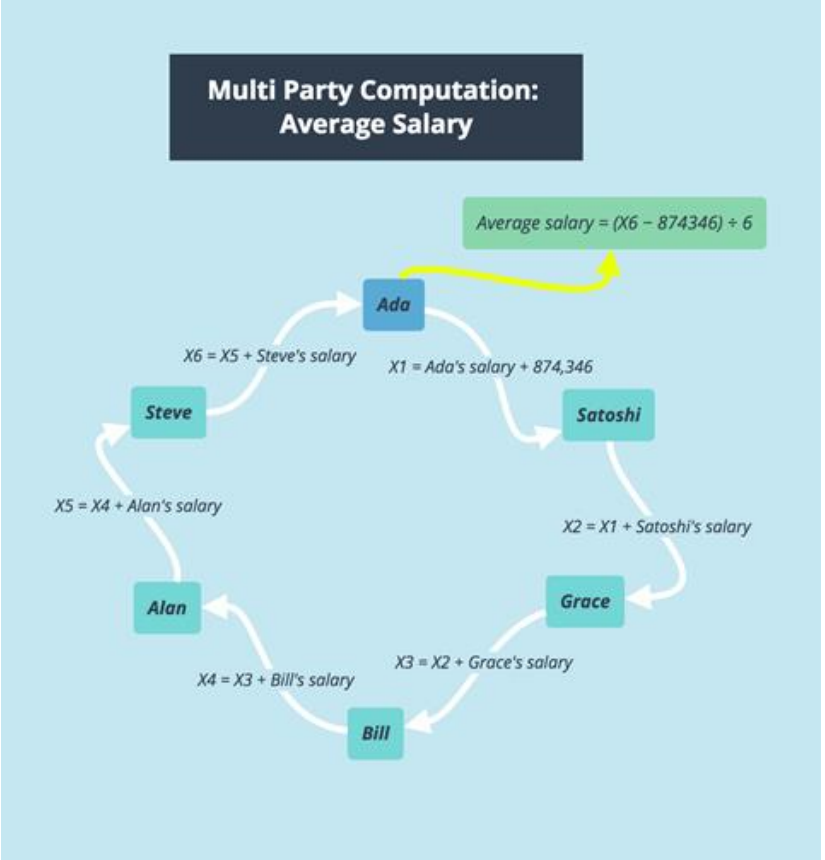
References: https://github.com/OpenMined/pysyft

Proposed Approach – PySyft (Federated Learning)



References: https://github.com/OpenMined/pysyft

Proposed Approach – PySyft (SMPC)



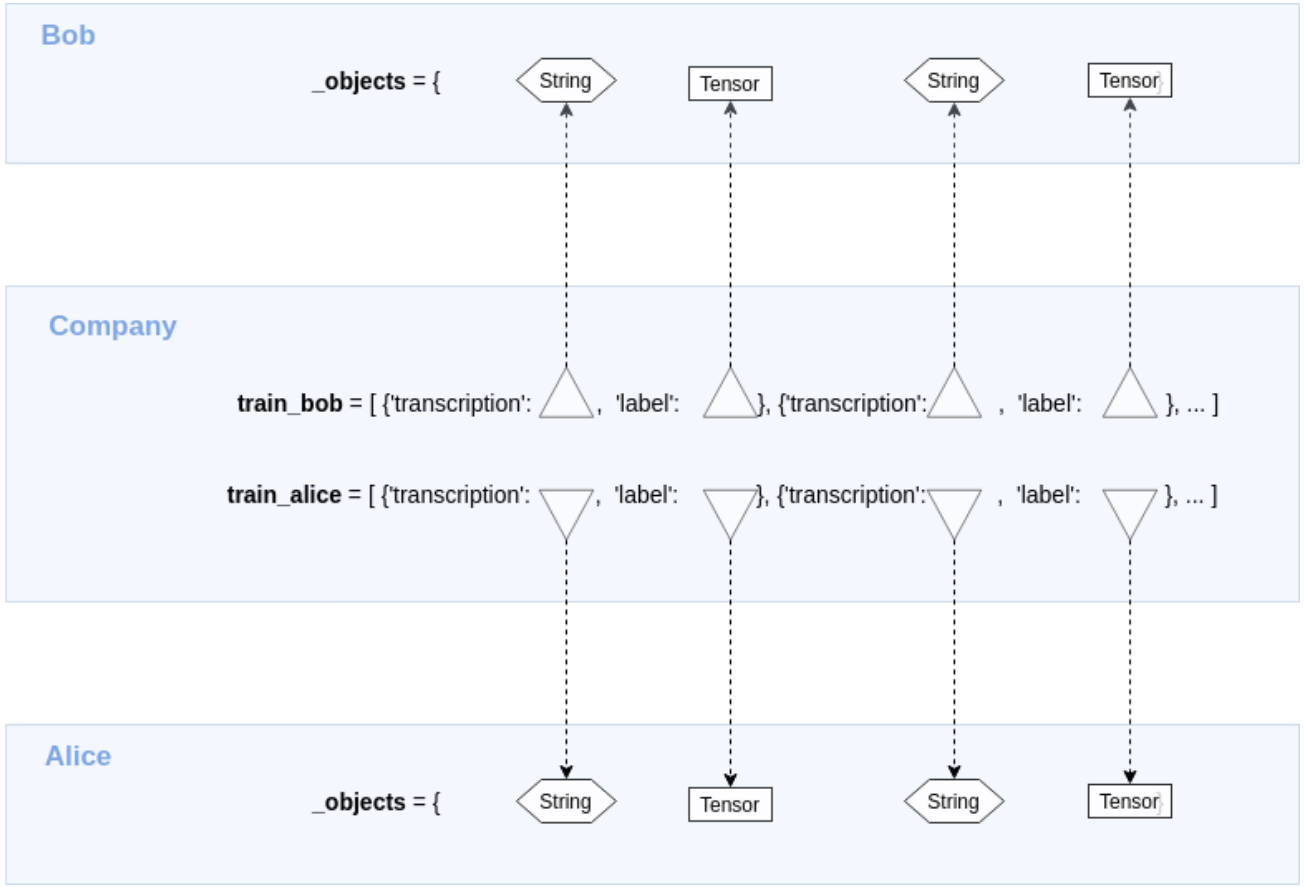
References: https://medium.com/wanchain-foundation/secure-multiparty-computation-and-shamirs-secret-sharing-on-wanchain-e502012b80ef

Proposed Approach – Virtual Environment

- A work environment is simulated with three main actors - a **company** and two clients owning **two private datasets** (Bob and Alice) but also a **crypto provider** that will provide the primitives for Secure Multi-Party Computation (SMPC).
- We simulate two private datasets owned by two clients (Bob and Alice) and distribute the respective datasets privately using a special ***share()*** function by the PySyft library.

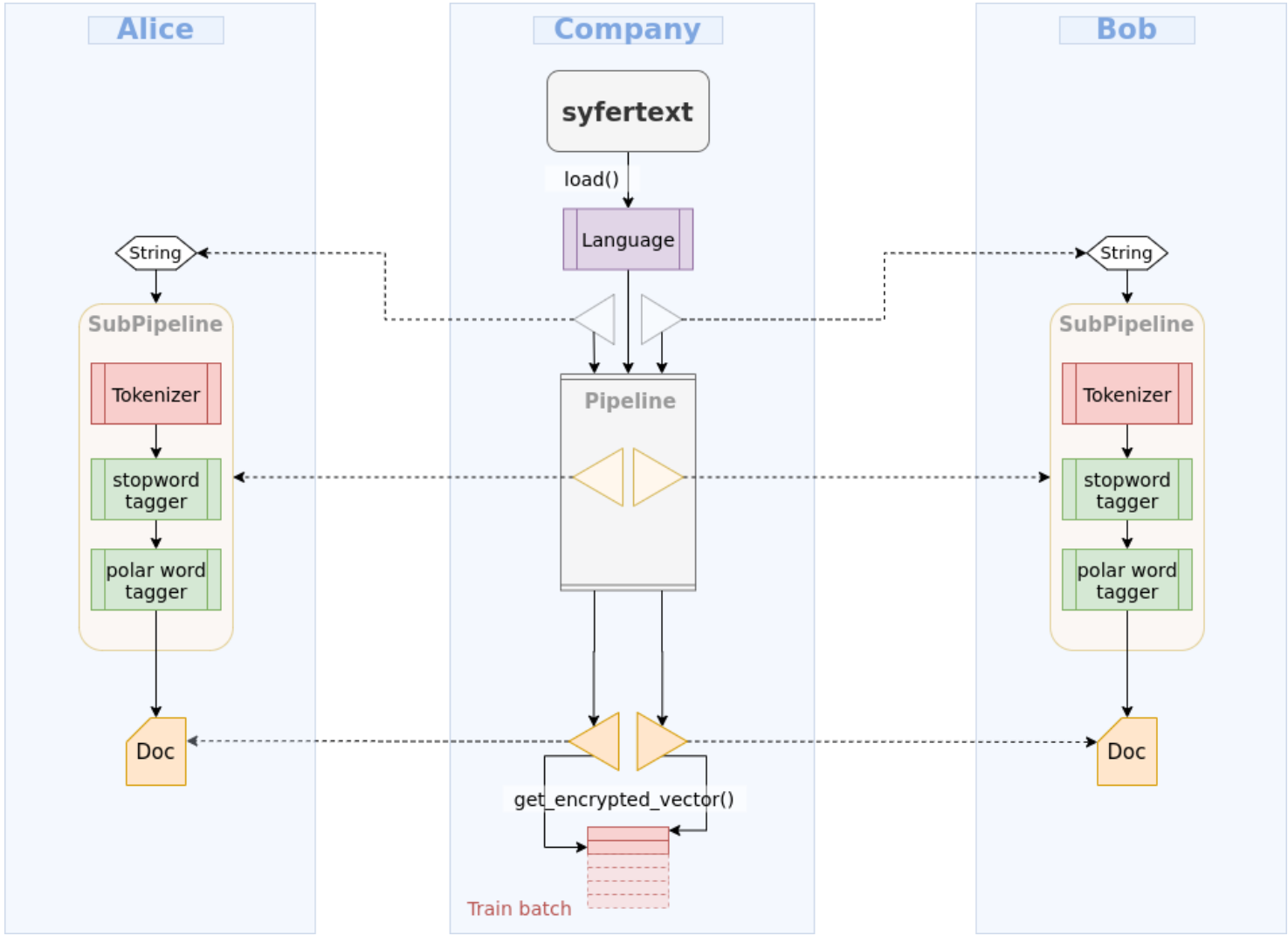
References: https://github.com/OpenMined/pysyft

Proposed Approach – SyferText (Distribute Dataset)



References: https://github.com/OpenMined/SyferText/blob/master/tutorials/usecases/UC01%20-%20Sentiment%20Classifier%20-%20Private%20Datasets%20-%20(Secure%20Training).ipynb

Proposed Approach – SyferText NLP Pipeline



References: https://github.com/OpenMined/SyferText/blob/master/tutorials/usecases/UC01%20-%20Sentiment%20Classifier%20-%20Private%20Datasets%20-%20(Secure%20Training).ipynb

Proposed Approach – Encrypted Classifier

- The hyper-parameters used for training and validation are as follows.
  - Embedding Dimension: The dimension of the embedding vector for the training dataset.
  - Batch Size: 128
  - Learning Rate: 0.001
  - Output classes: 4

```
Classifier (
    (fc1) Linear(in features=300, out features=128, bias=True)
    (fc2) Linear(in features=128, out features=64, bias=True)
    (fc3) Linear(in features=64, out features=32, bias=True)
    (fc4) Linear(in features=32, out features=16, bias=True)
    (fc5) Linear(in features=16, out features=2, bias=True)
)
```

Proposed Approach – Encrypted Deep Learning

- We create a hook for PyTorch to link it with PySyft to extend the functionalities of PyTorch so that we can use it for PySyft methods. We load the data, define our network structure, and share it across the virtual workers using a simple ***share()*** function in PySyft.

- Sending the tensors to virtual workers is as simple as calling the ***send(worker)*** method on the tensor. Any kind of remote operations can be performed on these tensors, in our case, we perform model training using forward and backward pass on these tensors.

- After the operations are performed we can call a simple ***get()*** function to return the tensor securely.

CODE – TRAINING AND HYPERPARAMETERS

Results – Accuracy and Loss

Training Accuracy	Validation Accuracy	Average Loss
78%	75%	27

Conclusion

- We observed that the model achieved around ***76% validation accuracy*** while the loss was reduced. However, we could not improve the model performance despite changing the model parameters as SyferText as of now has **limited support for different optimizers**.

- We assume that a different, deeper network architecture (RNN or LSTM) could potentially increase the model accuracy.

- Our goal to achieve an accuracy closer to the benchmark results is partially achieved. Ideally, there is always a trade-off between privacy and accuracy, especially when it comes to sensitive information the data and model privacy must be ensured at all costs.

QUESTIONS?