Amit Rastogi

Email: amitrastogi2206@gmail.com

Slack handle: @amit.rastogi

Abstract

Chatbot as a service using Secure Multiparty Computation

Project Showcase Challenge

**Title of the Project –** Chatbot as a service using Secure Multiparty Computation

**Background –** Chatbots provide a cost effective alternative to organizations for providing customer service. Organizations across a particular industry for e.g. banks could provide their chatbot model as a service as well as share their customer service interaction data with other organizations. To enable this kind of sharing however, a mechanism has to be there which allows both the chatbot model owner and the dataset owner to protect their model's IP and the privacy of their data respectively.

OpenMined PySyft library provides such a mechanism using constructs for Secure Multi-Party Computation which provides a mechanism to enable computation on encrypted data, enabling organizations to securely share their chatbot model and their customer service interaction data with other organizations.

**What is Secure Multi-Party Computation –** SMPC provides a mechanism to enable computation on encrypted data, without decrypting the underlying values themselves. It consists of private additive sharing and relies on the crypto protocols SecureNN and SPDZ.

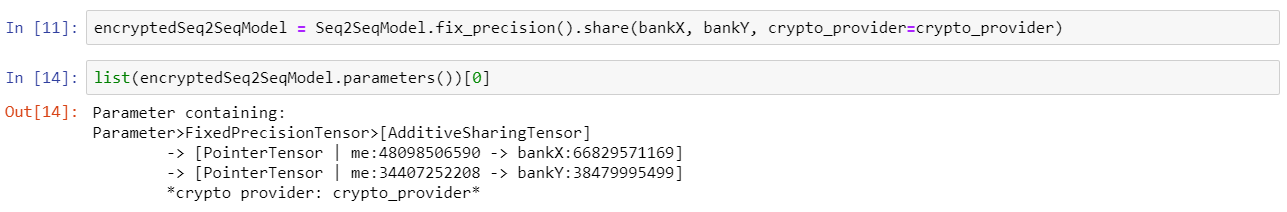
**Setup** **–** Suppose we have the following scenario: A large bank (say X) has created its own chatbot model by training it on its large customer service interaction dataset. Now consider a smaller bank (say Y) which does not have a large corpus of dataset of its own as well as computing resources to train its own chatbot model and hence approaches the larger bank to use their model.

Now bank X encrypts their model (say, a seq2seq neural network) and bank Y encrypts their data. Now both these banks could use these two encrypted assets to use the encrypted seq2seq model to make predictions on the encrypted bank Y data. Finally, the result of the prediction is sent back to bank Y in an encrypted way.

**PySyft’s Secure Multi-Party Computation (SMPC) components** **–** Now let us introduce some of the key components implemented in PySyft for using the SMPC protocol to perform encrypted computation over encrypted data.

1. Crypto provider – a secure worker responsible for reliably generating and distributing crypto primitives.
2. Fixed Precision Tensors – PySyft converts the PyTorch float tensors into fixed precision tensors using .fix\_precision() since SMPC uses crypto protocols which work on integers in finite fields.

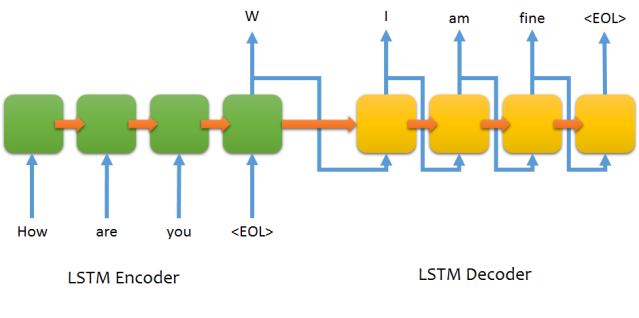
A typical sequence of api calls is shown below which can be used to encrypt and share the model between the participating entities:



1. Autograd Tensor – PySyft performs backpropagation and gradient updates even when working with encrypted integers in finite field due to a syft tensor called the AutogradTensor. The AutogradTensor stores the computational graph when operations are performed on encrypted values.
2. PySyft implements the crypto protocols SecureNN and SPDZ for secure computation.

**Project Objective** **–** The objective of this project is to demonstrate the use of a pre-trained seq2seq model for a chatbot to perform encrypted prediction using some sample text dataset using the PySyft’s SMPC primitives.

The seq2seq model intended to be used would have the following architecture-



**Steps involved in encrypted prediction using an encrypted model** –

1. Connect workers typically model owner, data owner and crypto provider
2. Build secret shares
3. Exchange secret shares
4. Send encrypted model to workers
5. Perform encrypted prediction using encrypted computation

**Implementation To Be Done** – Currently PySyft does not support full set of primitives to perform encrypted computation using a model having LSTM layers. This is work in progress and we plan to implement the same as soon as the primitives become available in PySyft.