**Federated Learning with Pretrained Text DNNs**

DATA590 Project Proposal

University of Washington Autumn 2019

Arjun Singh

Joel Stremmel

**Executive Summary / Introduction**

While machine learning on large datasets is the dominate paradigm in the field, there are a number of drawbacks to centrally aggregating data, namely privacy. Federated Learning aims to address this and has shown promise for [text completion tasks on mobile devices.](https://arxiv.org/pdf/1811.03604.pdf) The [Tensorflow Federated API](https://github.com/tensorflow/federated) provides methods to train Federated models and conduct Federated Learning experiments on data grouped by clients but never aggregated. Through our research partnership with Google, we aim to build on the existing body of Federated Learning experiments with a particular focus on fine-tuning text models for Natural Language Understanding.

**Problem Statement**

Federated Learning aims to train machine learning models in a distributed fashion without centralizing data but instead updating and passing model parameters from a central server to distributed entities and back to perform stochastic gradient descent. McMahan et al. propose the Federated Averaging algorithm in [Communication-efficient learning of deep networks from decentralized data.](https://arxiv.org/abs/1602.05629) This algorithm and associated experiments in the paper yield promising results but are limited to models with randomly initialized weights. We aim to address the problem of applying state-of-the-art pretrained text models like [BERT](https://arxiv.org/abs/1810.04805) (or more pruned versions, such as [ALBERT](https://arxiv.org/pdf/1909.11942.pdf)) for weight initialization in the context of the Natural Language Understanding tasks at which they excel by identifying training strategies for fine tuning language models on devices using Federated Learning.

**Background / Literature Review**

We reviewed a variety of papers to get up to speed on Federated Learning and pretrained text models including:

* List papers…

**Work-to-Date / Data Review**

**Data Streams**

Data for our research experiments is available via the [tff.simulation.datasets](https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets) module in the [Tensorflow Federated API](https://github.com/tensorflow/federated). The [tff.simulation.datasets.stackoverflow.load\_data()](https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow/load_data) method loads a mapping of clients (Stack Overflow user IDs) to examples (their posts and post metadata). The data is not brought into memory until training starts and is accessible in model-ready batches through the [tf.data.dataset](https://www.tensorflow.org/api_docs/python/tf/data/Dataset) module.

**Data Size and Attributes**

The data contains the full body text of all Stack Overflow questions and answers along with metadata. The API pointer is updated quarterly. The metadata includes:

* Creation date
* Question title
* Question tags
* Question score
* Type (Question or Answer)

The data is split into train, validation, and test sets with:

* Train: 342,477 distinct users and 135,818,730 examples.
* Validation: 38,758 distinct users and 16,491,230 examples.
* Test: 204,088 distinct users and 16,586,035 examples.

**Data Location**

The data is hosted by Kaggle and made available through the [tff.simulation.datasets](https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets) module in the [Tensorflow API](https://github.com/tensorflow/tensorflow). Stack Overflow owns the data and has released the data under the [CC BY-SA 3.0](https://creativecommons.org/licenses/by-sa/3.0/) license.

**Access Software**

The [Tensorflow Python API](https://github.com/tensorflow/tensorflow) provides access to the data. Using Python, we will able to load, explore, and construct models using the Stack Overflow dataset. The [tf.data.dataset](https://www.tensorflow.org/api_docs/python/tf/data/Dataset) module makes loading and training efficient by generating data samples in batches rather than requiring pulling the entire dataset into memory. This will enable analysis and model development using standard Python tools and the Google Colab notebook environment.

**Exploratory Analysis**

We have created a [GitHub repository](https://github.com/federated-learning-experiments/fl-text-models) with first-round exploratory analysis documented in the two PowerPoint presentations in the project documents directory. Additionally, we have run a number of the example notebooks from the Tensorflow Federated API, such as the [Shakespeare text generation notebook](https://github.com/tensorflow/federated/tree/master/docs/tutorials). We have started to adapt this example to work on the Stack Overflow data.

**Proposed Solutions**

**Languages and Framework**

Python

Tensorflow API

Google Cloud

**Tentative Pattern**

Fetch a pre-trained Wikipedia model as an “oracle model,” assuming that our experiments for fine-tuning strategies will apply to state-of-the art embedding approaches for text.

Fine tune on stack overflow through a series of experiments with model architecture, regularization, and Federated Averaging parameters.

Develop hypotheses on which parts of the model to fine-tune, and evaluate trade-offs between the number of clients used for training, the number of training rounds on device, the number of overall rounds, and the batch size.

**Final Deliverables**

Documented experiment parameters, results tables, and plots

Notebooks for replicating experiments

Presentation of findings

**Risks & Benefits of Proposed Solution**

**Schedule**

**Team Bios**