**Federated Learning with Pretrained Text DNNs**

DATA 590 Project Proposal

University of Washington, Autumn 2019

Arjun Singh

Joel Stremmel

**Executive Summary / Introduction**

While machine learning on large datasets is the dominant 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 enhancing text models for Natural Language Understanding tasks, such as Next Word Prediction, which falls in the realm of auto-regression and Next Sentence Prediction, which can be categorized in the realm of auto-encoding.

**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. However, they are currently trained on RNNs and LSTMs, without making use of the extremely powerful concept of [Attention in NLP](https://arxiv.org/pdf/1706.03762.pdf).

Our goal is to replicate the existing network architectures for Federated Averaging, stress testing their limits within our simulated environment in terms of compute, memory and power resources. We subsequently want to apply Attention based models for the currently established tasks, such as Next Word Prediction, and if successful in our pursuits, introduce the powerful Attention based model, BERT, and it’s pruned versions, into the realm of Federated Learning.

**Background / Literature Review**

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

1. H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Aguera y Arcas. “Communication-Efﬁcient Learning of Deep Networks." Accessed December 6, 2019. <https://arxiv.org/pdf/1602.05629.pdf>.
2. H. Brendan McMahan, Daniel Ramage, Kunal Talwar, Li Zhang. “Learning Differentially Private Recurrent Language Models.” Accessed December 6, 2019. <https://arxiv.org/pdf/1710.06963.pdf>.
3. Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, Jason Roselander. “Towards Federated Learning at Scale: System Design.” Accessed December 6, 2019. <https://arxiv.org/pdf/1902.01046.pdf>.
4. Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Francoise Beaufays Sean Augenstein, Hubert Eichner, Chloe Kiddon, Daniel Ramage. “Federated Learning for Mobile Keyboard Prediction.” Accessed December 6, 2019. <https://arxiv.org/pdf/1811.03604.pdf>.
5. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser. “Attention Is All You Need.” Accessed December 6, 2019. <https://arxiv.org/pdf/1706.03762.pdf>
6. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding.” Accessed December 6, 2019. <https://arxiv.org/pdf/1810.04805.pdf>.
7. Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut. “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.” Accessed December 6, 2019. <https://arxiv.org/pdf/1909.11942.pdf>.

We will include a comprehensive list of references with our final deliverables.

**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 include:

* 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 to pull 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**

We have three proposed solutions for the problem posed. Although we will aim to deliver all three, having more than one allows us to be more flexible with our target, depending on how long it takes to familiarize ourselves with NLP and FL concepts, the current TFF codebase, and the required duration of training and experimentation with various models:

1. Replicate the results from the current LSTM model published as part of the [GBoard paper](https://arxiv.org/pdf/1811.03604.pdf), that makes use of Federated Averaging for training a model for the task of Next Word prediction. The major goal is to assess what parts of this 1.4 million parameter model could be replaced or omitted, while trying to diminish the complexity/size of the network, without compromising on metrics like accuracy.
2. As an immediate next step in the enhancement of the aforementioned model, we aim to leverage Transformer models with Federated Averaging, for the task of Next Word prediction. These models are based on the paper, [“Attention Is All You Need”.](https://arxiv.org/pdf/1706.03762.pdf) Transformers, unlike many models based on Attention, can be Auto-Regressive and can be used for language modeling, as in NWP tasks. A [PyTorch implementation](https://github.com/pytorch/examples/blob/master/word_language_model/model.py) shared on [their website](https://pytorch.org/docs/stable/nn.html#torch.nn.Transformer) shall be used for reference.
3. Finally, if the above experiments are successful and if we have enough time, we’d like to assess the usage of pre-trained text deep neural networks such as [BERT](https://arxiv.org/abs/1810.04805) for “Federated Fine-Tuning”. This is a concept that hasn’t yet been explored, to the best of our understanding. As interesting and novel as this idea is, it is also equally hard to materialize due to the power/memory/compute constraints of personal mobile devices, the predominant target of Federated Learning, at the time of this writing. Well aware of these limitations, we also aim to explore light versions of BERT, such as [ALBERT](https://arxiv.org/pdf/1909.11942.pdf).

**Languages and Framework**

* Python
* Tensorflow API
* Google Cloud

**Final Deliverables**

* Documented experiment parameters, results tables, and plots
* Notebooks for replicating experiments
* Presentation of findings

**Risks & Benefits of Proposed Solution**

Major risk with BERT/ALBERT models: the task for which we’re training the first two of the three models listed above is Next Word Prediction (Language Modeling), which is best addressed by auto-regressive models. On the other hand, BERT and the light versions that succeeded it, are all auto-encoding based models. This implies that we will need to amend our task from NWP to something more along the lines of NSP (Next Sentence Prediction), a task on which BERT-type models have held the SOTA (state-of-the-art) scores for a long period of time since their advent.

**Schedule**

In the two and a half months we have to complete our project, we aim to tackle our three proposed solutions in order. Tentatively we aim to do the following:

January:

* Familiarize ourselves with the Stack Overflow data and the TFF dataset API for model training
* Replicate LSTM results from the GBoard paper using the Federated Averaging algorithm and work to reduce size and complexity of the LSTM without compromising much on accuracy

February:

* Identify main contributions from reducing LSTM size for Federated Averaging and finalize experiments with these methods
* Create a Transformer model with Federated Averaging
* Assess viability of BERT and ALBERT for federated fine tuning

March:

* Compile experimental results
* Create project deliverables
* March 11, 2020 is our final capstone presentation

**Team Bios**

**Joel Stremmel:** Senior Data Scientist at Optum and graduate student at University of Washington whose work and research are focused on using insights from data to improve healthcare at the individual and population level. Experienced at building machine learning pipelines for training and deploying disease progression models at scale. Interested in gaining hands-on experience with text modeling and distributed model training with Federated Learning.

**Arjun Singh:** I’m a Data Science graduate student at the University of Washington and am most interested in the domains of Deep Learning and Natural Language Processing, especially after a recent internship in that field in the Auto-ML team at Microsoft. With prior experience as a Quant at Goldman Sachs, and a researcher at a Healthcare Institute, I’m very passionate about letting data drive decision making.