The lead TA for this assignment is Debarati Das (das00015@umn.edu). Please communicate with her via Slack, email, or office hours.

This assignment requires 100% of programming. The goal of this assignment is to make sure you are able to build a simple neural net based text classifier using PyTorch from scratch.* I know you are scared if you haven't done any deep learning programming, but don't worry at all!

First, please carefully read tutorial slides/notebooks on Tutorial on Python programming and Tutorial on PyTorch programming taught by TA, Debarati Das, on Jan 25 and 30, respectively. Now that you have learned the basics of PyTorch programming through the tutorials and how to use Jupyter Notebooks in Google Colab, let's build a simple text classifier using PyTorch. In the tutorials, you developed a binary classifier for predicting whether a tweet is about a real disaster or not by using the Twitter Disaster dataset. In this assignment, you will implement **a two-layer multi-layer perceptron (MLP) text classifier using Pytorch on a new dataset**.

Step 1: Choose a dataset from TorchText.datasets

You can choose any dataset from PyTorch's torchtext. If you are using a laptop or local machine, choose a small dataset, such as IMDb and SST2. Below is an example script for loading the IMDB dataset.†.

```
# import datasets
from torchtext.datasets import IMDB

train_iter = IMDB(split='train')

def tokenize(label, line):
    return line.split()

tokens = []
for label, line in train_iter:
    tokens += tokenize(label, line)
)
```

Step 2: Stack two layers of MLP

You implemented a one-layer MLP classifier in the tutorial. This homework simply requires adding one more layer, as shown in the pseudo code below. Note that you added a 100-dimensional intermediate layer between the embedding layer and the last class layer.

```
class MLP(nn.Module):
    def __init__(self, vocab_size, embed_dim, num_class):
        super().__init__()
        self.embedding = nn.EmbeddingBag(vocab_size, embed_dim, sparse=True)
        num_layer = 100
        # self.fc = nn.Linear(embed_dim, num_class)
        self.fc1 = nn.Linear(embed_dim, num_layer)
        self.fc2 = nn.Linear(num_layer, num_class)

        # initialize the weights
        self.init_weights()

    def init_weights(self):
```

*https://pytorch.org/
†http://ai.stanford.edu/~amaas/data/sentiment/

```
        ..

    def forward(self, text, offsets):
        embedded = self.embedding(text, offsets)
        return self.fc2(self.fc1(embedded))
```

Step 3: Training and Analysis

Following the PyTorch tutorial, you can start training your model after loading a dataset and designing a classifier model. On the test set, what is the accuracy? What is the performance of a two-layer MLP compared to a single-layer MLP? Do you have a look at the test set samples that are incorrectly predicted by the model? Why are the errors occurring? This assignment will be the basis of your next assignment and class project that require more advanced programming, analysis, and deep learning knowledge.

Deliverable

This assignment has no credit. Jupyter notebooks in Colab can be submitted but they are completely optional. The only thing you have to submit is **your team name and members** for future assignment and class project. Please create a group of your team members on Canvas and submit this homework as a group to Canvas by Jan 30, 11:59pm.

References