COSC 480C (NLP) Homework 3: Fall 2023

The due date for this homework is **Monday, November 13, 11pm ET**. Start early!

Introduction

This assignment is designed to guide you through the implementation and evaluation of a neural language model using a simple recurrent neural network.

Guiding Question

- How do we build neural networks?
- How do we train a neural language model?
- How do we evaluate our models?

Learning Objectives

- Implement a recurrent neural network language model
- Train a neural language model
- Understand how to tune neural network hyperparameters
- Evaluate a pre-trained neural language model
- Understand how core concepts of language modeling are transferred to a neural network setting

Rubric

1. Part 1 - 100 Total Points

Question	Points
RNNCell init	10 points
RNNModel init	10 points
RNNCell forward	40 points
RNNModel forward	40 points

2. Part 2 - 50 Total Points

Question	Points
train	50 points

3. Part 3 - 50 Total Points

Your assignment

Your task is to complete following steps:

1. Clone the homework repository

   ```
   git clone https://github.com/DavisLingTeaching/HW3-RNNs.git
   ```

2. Install pytorch (link)
3. Complete `model.py` described in Part 1
4. Complete `train.py` described in Part 2
5. Complete `ling.py` described in Part 3
6. Submit your completed programs to Moodle.

Part 1 - Implementing

Building a RNN language model

In this portion of the assignment you will implement a recurrent neural network language model. A description of the function and samples of its use are given in the function docstring. **Please read the docstrings**. I have implemented some other functions that you may find useful!

Your task is broken into two components: implementing an RNN cell and implementing the full model. The cell handles the computation for a single layer, while the full model handles the full forward pass from input to output through layers of RNN cells.

Note that the input to your RNN model are token ids. In addition to the RNN cell layers, you should have an embedding matrix whose weights are learned. Additionally, you should have a decoder (with no bias) that maps from the output of the final RNN cell to the vocab space. This mirrors the E and V matrices, respectively, in the RNN handout from class. The handout and its key are provided in the repo under notes.

Testing your RNN language model

Some (non-exhaustive) tests can be run by using test.py.

For example,

```
python test.py --test init_cell
```

will test some aspects of the init of your RNNCell

Part 2 - Training

Training a RNN language model

In this portion you will train your RNN with some data. Starter code is included with train.py that handles batching the data. **Please review the existing code**. You should review the feedforward neural network lecture notes (link).

Part 3 - Evaluating

In this portion you will evaluate a pre-trained model for linguistic knowledge. More details will be added later.

Good luck :) !