1 Instructions

Please note the following guidelines for submitting your assignments:

1. Use Colab to submit your assignments.
2. You may use any machine learning library (pyTorch, Tensorflow, etc.) to implement this assignment
3. Share the final version of your assignments with the following email ID:
   (a) ramaseshan.ta@gmail.com
4. I will NOT run/change your Python notebook.
5. Naming Conventions for the assignments:
   (a) The first part of the filename should be your First name.
   (b) The second part of the filename should be your roll number.
   (c) The third part of your assignment should be Assignment0X, where X is the assignment number.

   Example: `SriramBMC202204_Assignment01.ipynb`
6. Make sure that all the results are available when you share them.
7. **Optional**: You may use Github to store your versions of the assignment. Advantages - You will never lose your code if you check-in the code into the GitHub repository after changes regularly.

2 Assignment 1 (A1) - Parts of Speech Tagging with Recurrent Neural Networks (RNNs)

3 PoS Tagging using RNN

3.1 Objective

In this assignment, you will explore the task of parts-of-speech (POS) tagging using Recurrent Neural Networks (RNNs). You will implement and train an RNN-based model to automatically assign POS tags to words in a given sentence.

3.1.1 Learning Outcomes

- Understand the concepts of POS tagging
- Learn the importance of PoS tagging
- Gain practical experience in training and building models using RNN for sequence labeling
- Analyze the performance - in terms of time and space complexity and accuracy of labeling by randomly checking the inferred tags

3.1.2 Dataset

WSJ dataset containing sentences and their corresponding POS tags is attached. The attached compressed file contains around 200 raw ($\approx$ 79K words) and tagged files($\approx$ 79K words + $\approx$ 15K tags). You may split the dataset into training(80%), validation(10%), and testing sets(10%).

3.1.3 Model Implementation

Implement an RNN-based model for POS tagging, such as an LSTM or GRU network. Using your Machine library of your choice, define the network architecture, including input layer, embedding layer, recurrent layer(s), and output layer and finally choose appropriate activation functions and loss functions for the task.

3.1.4 Model Training

Train the model on the training data using an appropriate optimizer (e.g., Adam). Monitor the training progress by evaluating the model's performance on the validation set during training. Use techniques like early stopping and regularization to prevent overfitting.

3.1.5 Evaluation and Analysis

Evaluate the final model's performance on the testing set using standard metrics like accuracy, precision, recall, and F1-score for each POS tag. Analyze the results and identify any potential errors or areas for improvement.

3.1.6 Experimentation (Optional)

- Try different RNN architectures or hyper-parameters to improve the model's performance
- Explore methods to handle unknown words or out-of-vocabulary (OOV) tokens

3.1.7 Deliverables

A Jupyter notebook or Python script. A report summarizing your findings, including model architecture, training details, evaluation results, and analysis of errors and potential improvements. A few lines would suffice

3.1.8 Format found in the corpus

Raw	Trinity Industries Inc. said it reached a preliminary agreement to sell 500 railcar platforms to Trailer Train Co. of Chicago. Terms weren't disclosed. Trinity said it plans to begin delivery in the first quarter of next year.
Tagged	Trinity/NNP Industries/NNPS Inc./NNP said/VBD it/PRP reached/VBD a/DT preliminary/JJ agreement/NN to/TO sell/VB 500/CD railcar/NN platforms/NNS to/TO Trailer/NNP Train/NNP Co./NNP of/IN Chicago/NNP ./. Terms/NNS were/VBD n't/RB disclosed/VBN ./. Trinity/NNP said/VBD it/PRP plans/VBZ to/TO begin/VB delivery/NN in/IN the/DT first/JJ quarter/NN of/IN next/JJ year/NN ./.