Assignment 3: Language Modeling

Homework assignments will be done individually: Each student must hand in their own answers. Use of partial or entire solutions obtained from others or online is strictly prohibited. Electronic submission on Canvas is mandatory.

You will need to use the Penn Treebank corpus for this assignment. Three data files are provided: train.txt, valid.txt, and input.txt. You can use train.txt to train your models and use valid.txt for testing. File input.txt can be used for a sanity check on whether the model produces coherent sequences of words for unseen data with no next word.

1. **N-gram** (60 points)
   (a) (0 pts) Run the functions including remove punctuation, remove url, remove number, lowercase and tokenize.
   (b) (10 pts) Calculate the unigram and bigram count in the training dataset.
   (c) (5 pts) Implement a BiGram model for language modeling. Fill in the code for class `BiGram`.
   (d) (15 pts) Implement Good Turing smoothing. Fill in the code in the class `GoodTuring`. Hint:
   - Use power law to replace empirical N_c when c is greater than 100. (See page 69 in lecture slides of language modeling)
   - You will need to calculate frequency of frequency for both bi-gram terms and unigram terms. $P_{GT}(w_2|w_1) = \frac{c^*(w_1,w_2)}{c(w_1)}$

   (e) (15 pts) Implement Kneser-Ney smoothing using:

$$P_{\text{KN}}(w_i|w_{i-1}) = \frac{\max(c(w_{i-1},w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1})P_{\text{CONTINUATION}}(w_i)$$

   where

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})}|\{w : c(w_{i-1},w) > 0\}|$$

$$P_{\text{CONTINUATION}}(w) = \frac{|w_{i-1} : c(w_{i-1},w) > 0|}{\sum_{w'}|\{w'_{i-1} : c(w'_{i-1},w') > 0\}|}$$

   Check the slides for specifying the value of d. Fill in the code for class `KneserNey`.
   (f) (10 pts) Implement Perplexity and use perplexity to evaluate BiGram, Good-turing, and Kneser-Ney. Fill in the code for function perplexity.
   (g) (5 pts) Randomly select the 10 lines in the input.txt file and print the predictions of next words using your BiGram, Good-Turing, and Kneser-Ney models. Implementing the prediction function.

2. **RNN** (35 points) You can use libraries such as PyTorch or TensorFlow for this part.
   (a) (0 pts) Data preparation. Split the tokens into feature and label.
   (b) (5 pts) Padding the sentences in a batch with equal lengths.
   (c) (10 pts) Build your RNN model. The model should include an embedding layer (input layer using word embedding vectors), a hidden layer (RNN cells), an output layer (predict next word).

(d) (5 pts) Implement the sequence to sequence loss.

(e) (10 pts) Model evaluations: Prove that perplexity is the exponential of the total loss divided by the number of predictions. Calculate the perplexity score of your model predictions.

(f) (5 pts) Print the predictions of next words using the RNN model for the same 10 lines of input.txt as in N-gram. Implement the prediction.

3. **Conclusion** (5 points) Briefly analyze the result of N-Gram and RNN.

4. **Submission Instructions** You shall submit a zip file named Assignment3_LastName_FirstName.zip which contains:

   - code files (.ipynb or/and .py) including all the code, comments, plots, and result analysis. You need to provide detailed comments in English.
   - If you submit your code in py files, you can include a pdf file to show your plots and result analysis.