COMP4650/6490 Document Analysis Assignment 3 – NLP

For this assignment, you will implement several common NLP algorithms: Interpolated language models, dependency parsing, and probabilistic context free grammars in Python.

Throughout this assignment you will make changes to the provided code. You should submit a .zip file containing all of your python files, **BUT NOTHING ELSE.**

For this assignment you are only being marked on your coding solutions. You will lose marks if your code is inefficient, difficult to read, does not run on our platform, or is incorrect.

Question 1: Language Models (25%)

The file *language_models.py* contains code for constructing an absolute discounting bigram interpolation model. Your task is to implement the KNSmoothing class to compute Kneser-Ney smoothing of bigrams, according to the formulas given in the lecture slides. You should follow the structure of the provided AbsDist class where possible but modify it to use the Kneser-Ney formulas.

Question 2: Dependency Parsing (25%)

The file *dependency_parser.py* contains a partial implementation of Nivre's arc-eager dependency parsing algorithm. Your task is to implement the missing *reduce* and *right_arc* functions. Your implementation should follow the transitions as given by the lecture slides. Make sure that your functions return True if the operation was successfully applied, or else False.

Question 3: Probabilistic Context Free Grammars (50%)

For this question you will be using *syntax_parser.py.* You have been provided with a training dataset containing 500 sentences from a made-up language (train_x.txt), which is given by a probabilistic context free grammar (PCFG). Each of the sentences in the training dataset is labelled with its correct syntax (train_y.txt), that is the parse tree that derived this sentence. Your task is to estimate the PCFG that generated the sentences. You should estimate the probability of a transition A -> B by counting the number of times that the transition A -> B occurs in the training dataset, and dividing by the number of times A occurs.

When your *syntax_parser.py* code is run, it should print a list of all of the transition rules that occur in the grammar, along with the probability of each one.