CMPT 413/713: Natural Language Processing **(Due: 10/26/21)**

Programming Homework #3

Group Members: Joshua Malmberg, Hewei Cao, Keenan Byun
Group Name: Mochi

1 Objective

Phrasal chunking is the task of partitioning a sentence into groups called phrasal chunks. Each phrasal chunk corresponds to an atomic syntactic group of words, ie. splitting the group up further would separate words which play the same syntactic role in a sentence. With a naïve phrasal chunking model, spelling mistakes and other forms of noise in the input are likely to lead to out-of-vocabulary words which significantly decrease the model performance. Robust phrasal chunking extends includes measures to account for noisy input and ensure performance. Given a naïve phrasal chunking model, our goal in this assignment is to implement a robust phrasal chunking algorithm.

2 Method

To achieve robust phrasal chunking, we implemented three models and compared their relative performance.

The first model expands the LSTM RNN in the default solution by expanding the input to include each words character-level representation in addition to the word embeddings. The character-level representation takes the form of a vector with three sub-vectors. The first and third sub-vectors are one-hot encodings of the first and last characters of the word, respectively; whereas, the second sub-vector is a bag-of-characters vector representing the interior characters of the word.

The second model adds a second LSTM RNN. The hidden states of the new RNN is concatenated with the word embeddings as the input to the tagger RNN provided in the default solution. This new RNN takes as input the character-level representations of the words in the sentence and stores encodes information about them in the hidden state. The character-level representations used are similar to those in the first model, except that the second sub-vector is replaced with two vectors: one for the interior vowels of the word and one for the interior consonants of the word.

The third model implements an scRNN to remove noise in the input words before they are fed to the tagger RNN. The new RNN added takes as inputs the same character-level representations used in the first model. To train the model, noise is added to a set of words which are fed as training data to the RNN. The cross-entropy loss was then computed by comparing the RNN predictions with the actual words, and the loss was backpropagated to tune the RNN.

3 Results

	Dev Score
Method 1	75.9
Method 2	75.13
Method 2 w/ expanded CLR	76.98
Method 3	na

The best performance was obtained using method 2. A working version of method 3 was implemented, however the scores were very poor, indicating an issue with the training algorithm.

4 Contributions

Hewei Cao: Implemented Method 1 and Contributed to Jupyter Notebook; **Keenan Byun:** Implemented Method 2 and contributed to Jupyter Notebook; **Joshua Malmberg:** Implemented Method 3 and Wrote Report.