COSI134 Project 4: Train a neural network parser

Due: December 17, 2020

In the fourth and final project, you are asked to train a neural network parser using the Penn TreeBank data. You are asked to use the encoder-decoder framework and experiment with the various attention mechanisms to observe their effect on the performance of the parser. You are asked to work with the starter code that has been provided, although you are free to modify the code as you see fit.

The provided code implements the seq2seq model based on RNN, and one core idea that makes seq2seq models successful is attention. Recall that attention has a few variants. One of them is called the Bahdanau attention, an additive attention mechanism named after its inventor:

$$\psi_\alpha(m,n) = \boldsymbol{v}_\alpha \cdot \tanh(\Theta_\alpha[\boldsymbol{h}_m^{(t)} + \boldsymbol{h}_n^{(s)}])$$

Another one is the mulpilative attention called the Leong attention, also named after its inventor:

$$\psi_\alpha(m,n) = \boldsymbol{h}_m^{(t)\top}\Theta_\alpha\boldsymbol{h}_n^{(s)}$$

The two attention mechanisms differ in how they compute the interaction scores between the target query and the source key. There is also a version of the Leong attention that does not use weight matrix between the target query and the source key:

$$\psi_\alpha(m,n) = \boldsymbol{h}_m^{(t)\top}\boldsymbol{h}_n^{(s)}$$

In the project, your asked to implement the two variants of th xperiment with how they perform against the baseline that does not use the attention

Implementation tips:

- Like the POS tagging project, it's advisable to start small and make sure your code works on a smaller data sample.
- Preprocessing: Before you can use a sequence-to-sequence model to perform syntactic parsing, you first need to linearize the parsing trees when you prepare the training data. The trees need to be linearized in a way that can be mapped back to well-formed parse trees. It is also important to bear in mind that you need to reduce the size of the output "vocabulary" as much as possible to make the model practical. The preprocessing code has been provided, and you just need to call the right functions to produce linearized trees for training.

Experiments

You are asked to experiment with different attention mechanisms and observe their effects on parser performance.

- Experiment 1: Train a baseline model that does not use any attention mechanism. Tune the hyperparameters to get the best result you can, and evaluate the results with the Bleu metric.
- Experiment 2: Train a model with the first variant of Leong attention. Tune the hyperparameters to get the best result you can, and evaluate the results with the Bleu metric.

- Experiment 3: Train a model with the second variant of Leong attention. Tune the hyperparameters to get the best results possible, and evaluate the results with the Bleu metric.

Evaluation

Evaluation with Bleu:

With a seq2seq parsing model, you produce linearized trees, and there is no guarantee that the output of the model is well-formed. Therefore, we use Bleu, a metric for Machine Translation. Bleu works by comparing the correct sequence and the output sequence. Bleu may not be the ideal for evaluating parsing, but it gives you a sense of how your model works. The reason we use Bleu is that it does not require the parsing output to be well-formed. The Bleu implementation can be found on Github: https://github.com/mjpost/sacrebleu (https://github.com/mjpost/sacrebleu). A wrapper is included in 'utils.py'.

Evaluation with Evalb:

The standard evaluation metric for evaluating parsing is bracketed precision, recall, and F1-score, and the standard software that researchers use to evaluate constituent parsing is called "evalb". The original version of the software can be found here: https://nlp.cs.nyu.edu/evalb/ (https://nlp.cs.nyu.edu/evalb/), and this software is included in this package. There are also java reimplementations of the software at the Stanford Core NLP. The software outputs many metrics, but the main metrics are labeled precision and labeled recall, which are based on counting the number of matching constituents between the gold tree and the system output. For bonus points, you can optionally tr do this, you need to do some postprocessing to map the he output of your model is well-formed (e.g., adding bracket tool will not work.

Report

Your write-up should be no more than three pages. In your w

- give a brief description on your code structure.
- give a description of your model, including the model architecture, hyperparameters, etc.
- report experiments and results with different attention mechanisms

In [ ]: