COSI134 Project 4: Train a neural network parser

Due: December 17, 2019

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 the starter code that has been provided, although you are free to modify the code as you see fit.

Like the POS tagging project, it's advisable to start small and make sure your code works on a smaller data sample. The output of your parser needs to be evaluated with a standard software 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 there is a version of this software implemented on our servers (/home/j/clp/chinese/bin/evalb). 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 parser tree and the system output.

Some implementation tips:

- 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.
- Postprocessing: The output of the decoder will be a linearized sequence that represents the parse of the input sentence. In order to evaluate the parse, it needs to be mapped back to tree structure in the original PTB format. With a sequence-to-sequence model, there is no guarantee that the output parses are well-formed, with matching parentheses, etc. So you need to do some (automatic) postprocessing to prepare the trees for evaluation.

Experiments

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

Report

Your write-up should be no more than three pages. In your write-up, you need to:

- 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 [ ]:
```
import nltk.corpus
reader = nltk.corpus.BracketParseCorpusReader(r'./train/', r'.*/wsj_.*\.mrg')
```

In [2]:
```
reader.sents()    # gives you plain sentences
```

Out[2]:
```
[['In', 'an', 'Oct.', '19', 'review', 'of', '``', 'The', 'Misanthrope', "''", 'at', 'Chicago', "'s",
'Goodman', 'Theatre', '-LRB-', '``', 'Revitalized', 'Classics', 'Take', 'the', 'Stage', 'in', 'Wind
y', 'City', ',', "''", 'Leisure', '&', 'Arts', '-RRB-', ',', 'the', 'role', 'of', 'Celimene', ',',
'played', '*', 'by', 'Kim', 'Cattrall', ',', 'was', 'mistakenly', 'attributed', '*-2', 'to', 'Christ
ina', 'Haag', '.'], ['Ms.', 'Haag', 'plays', 'Elianti', '.'], ...]
```

In [3]:
```
reader.parsed_sents()    # gives you gold parsed trees
```

Out[3]:
```
[Tree('S', [Tree('PP-LOC', [Tree('IN', ['In']), Tree('NP', [Tree('NP', [Tree('DT', ['an']), Tree('NN
P', ['Oct.']), Tree('CD', ['19']), Tree('NN', ['review'])]), Tree('PP', [Tree('IN', ['of']), Tree('N
P', [Tree('``', ['``']), Tree('NP-TTL', [Tree('DT', ['The']), Tree('NN', ['Misanthrope'])]), Tree
("''", ["''"]), Tree('PP-LOC', [Tree('IN', ['at']), Tree('NP', [Tree('NP', [Tree('NNP', ['Chicag
o']), Tree('POS', ["'s"])]), Tree('NNP', ['Goodman']), Tree('NNP', ['Theatre'])])])])]), Tree('PRN',
[Tree('-LRB-', ['-LRB-']), Tree('``', ['``']), Tree('S-HLN', [Tree('NP-SBJ', [Tree('VBN', ['Revitali
zed']), Tree('NNS', ['Classics'])]), Tree('VP', [Tree('VBP', ['Take']), Tree('NP', [Tree('DT', ['th
e']), Tree('NN', ['Stage'])]), Tree('PP-LOC', [Tree('IN', ['in']), Tree('NP', [Tree('NNP', ['Wind
y']), Tree('NNP', ['City'])])])])])]), Tree(',', [',']), Tree("''", ["''"]), Tree('NP-TMP', [Tree('N
N', ['Leisure']), Tree('CC', ['&']), Tree('NNS', ['Arts'])]), Tree('-RRB-', ['-RRB-'])])])]), Tree
(',', [',']), Tree('NP-SBJ-2', [Tree('NP', [Tree('NP', [Tree('DT', ['the']), Tree('NN', ['role'])]),
Tree('PP', [Tree('IN', ['of']), Tree('NP', [Tree('NNP', ['Celimene'])])])])]), Tree(',', [',']), Tree
('VP', [Tree('VBN', ['played']), Tree('NP', [Tree('-NONE-', ['*'])]), Tree('PP', [Tree('IN', ['b
y']), Tree('NP-LGS', [Tree('NNP', ['Kim']), Tree('NNP', ['Cattrall'])])])]), Tree(',', [','])]), Tre
e('VP', [Tree('VBD', ['was']), Tree('VP', [Tree('ADVP-MNR', [Tree('RB', ['mistakenly'])]), Tree('VB
N', ['attributed']), Tree('NP', [Tree('-NONE-', ['*-2'])]), Tree('PP-CLR', [Tree('TO', ['to']), Tree
('NP', [Tree('NNP', ['Christina']), Tree('NNP', ['Haag'])])])])]), Tree('.', ['.'])]), Tree('S', [Tr
ee('NP-SBJ', [Tree('NNP', ['Ms.']), Tree('NNP', ['Haag'])]), Tree('VP', [Tree('VBZ', ['plays']), Tre
e('NP', [Tree('NNP', ['Elianti'])])]), Tree('.', ['.'])]), ...]
```

In [4]:
```
reader.parsed_sents()[0].__str__()    # gives you the tree represented in a string. Make sure to remo
ve the newline signs before training.
```

Out[4]:
```
"(S\n  (PP-LOC\n    (IN In)\n    (NP\n      (NP (DT an) (NNP Oct.) (CD 19) (NN review))\n      (PP\n
(IN of)\n        (NP\n          (`` ``)\n          (NP-TTL (DT The) (NN Misanthrope))\n          ('' 
'')\n          (PP-LOC\n            (IN at)\n            (NP\n              (NP (NNP Chicago) (POS
's))\n              (NNP Goodman)\n              (NNP Theatre)))))\n      (PRN\n        (-LRB- -LRB
-)\n        (`` ``)\n        (S-HLN\n          (NP-SBJ (VBN Revitalized) (NNS Classics))\n
(VP\n            (VBP Take)\n            (NP (DT the) (NN Stage))\n            (PP-LOC (IN in) (NP
(NNP Windy) (NNP City)))))\n        (, ,)\n        ('' '')\n        (NP-TMP (NN Leisure) (CC &) (NNS
Arts))\n        (-RRB- -RRB-))))\n  (, ,)\n  (NP-SBJ-2\n    (NP (NP (DT the) (NN role)) (PP (IN of)
(NP (NNP Celimene))))\n    (, ,)\n    (VP\n      (VBN played)\n      (NP (-NONE- *))\n      (PP (IN
by) (NP-LGS (NNP Kim) (NNP Cattrall))))\n    (, ,))\n  (VP\n    (VBD was)\n    (VP\n      (ADVP-MNR
(RB mistakenly))\n      (VBN attributed)\n      (NP (-NONE- *-2))\n      (PP-CLR (TO to) (NP (NNP Ch
ristina) (NNP Haag)))))\n  (. .))"
```