1. Neural Transition-Based Dependency Parsing

b A parsing process of a sentence "I parsed this sentence correctly":

Stack	**Buffer**	**New dependency**	**Transition**
[ROOT]	[I, parsed, this, sentence, correctly]		Initial Configuration
[ROOT, I]	[parsed, this, sentence, correctly]		SHIFT
[ROOT, I, parsed]	[this, sentence, correctly]		SHIFT
[ROOT, parsed]	[this, sentence, correctly]	parsed → I	LEFT-ARC
[ROOT, parsed, this]	[sentence, correctly]		SHIFT
[ROOT, parsed, this,sentence]	[correctly]		SHIFT
[ROOT, parsed, sentence]	[correctly]	sentence → this	LEFT-ARC
[ROOT, parsed]	[correctly]	parsed → sentence	RIGHT-ARC
[ROOT, parsed,correctly]	[]		SHIFT
[ROOT, parsed]	[]	parsed → correctly	RIGHT-ARC
[ROOT]	[]	ROOT → parsed	RIGHT-ARC

c Each word must be added to stack, i.e. it's n SHIFT steps.
Each word must be dependent on other word, i.e. it's n RIGHT/LEFT-ARC steps.
One step for connecting to the ROOT. In total 2n+1 steps.

g Training results:
dev UAS: 88.45
test UAS: 88.83

h

1. "It is on loan from a guy named Joe O'Neill in Midland , Texas"
   Error Type: Prepositional Phrase Attachment Error
   Incorrect dependency: named → Midland
   Correct dependency: guy → Midland

2. "Brian has been one of the most crucial elements to the success of Mozilla software ."
   Error Type: Modifier Attachment Error
   Incorrect dependency: elements → most
   Correct dependency: crucial → most

3. "I was heading to a wedding fearing my death"
   Error Type: Verb Phrase Attachment Error
   Incorrect dependency: wedding → fearing
   Correct dependency: heading → fearing

4. "It makes me want to rush out and rescue people from dilemmas of their own making."
   Error Type: Coordination Attachment Error
   Incorrect dependency: makes → rescue
   Correct dependency: rush → rescue

2. Syntactic Parsing

(a) The long sentences are caused by the rules $NP \rightarrow NP\ PP$ and $PP \rightarrow Prep\ NP$, as these rules can apply recursively an infinite number of times.

In particular, the first rule $NP \rightarrow NP\ PP$ contains the nonterminal symbol NP on both the left and right hand sides, so it may be applied an infinite number of times.

The second rule can also be used to create infinite recursion by expanding NP to $NP\ PP$ (rule 1) to $NP\ Prep\ NP$ (rule 2),

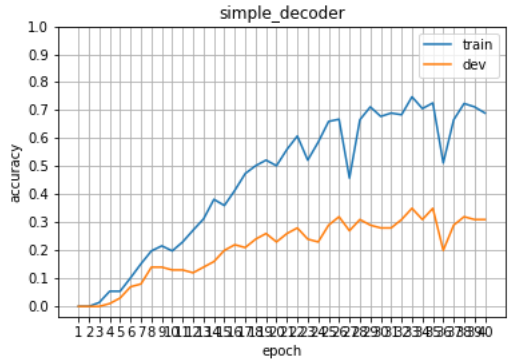
(b) At each iteration of the generation algorithm, a nonterminal is expanded by choosing a random expansion in proportion to their weights. The preterminal symbol $Noun$ has six expansions in the grammar, each of weight one ($Noun \rightarrow Adj\ Noun$, $Noun \rightarrow president$, $Noun \rightarrow airline$, etc.). Therefore every noun has only probability 1/6 of being preceded by at least one adjective, probability 1/36 of being preceded by at least two adjectives, and so forth (geometric distribution).

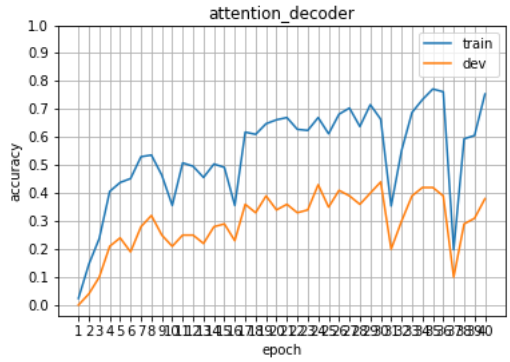
(c) The problem in (a) could be fixed by reducing the weight of the rule $NP \rightarrow NP\ PP$ (or increasing the weights of all other rules expanding NP). The problem in (b) could be fixed by increasing the weight of the rele $Noun \rightarrow Adj\ Noun$ (or decreasing the weight of all other rules expanding preterminal $Noun$).

(d) (implemented in cky.py)

(e) Output parse: (S (NP (Det the) (Noun president)) (VP (Verb ate) (NP (Det the) (Noun (Adj delicious) (Noun sandwich)))))

3. Semantic Parsing




attention_decoder
train
dev
accuracy
epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0