CSE 325/425 (Spring 2021) Homework 4

Due on 11:55pm, Apr 7, 2021

Grading: All questions have the same points (25 each). We will randomly grade some of the questions.

Submitting: Only electronic submissions on Coursesite are accepted. You can handwrite your answers on papers and then scan them to images. If you need to plot figures using a computer, the plotted files should be saved and included in the submitted pdf file. Submit a single pdf file named

<Your LIN>HW4.pdf

Other format will not be accepted.

Questions:

1. Modify the codes for RNN in the associated IPython notebook "Recurrent Neural Networks.ipynb" to remove the tanh activation function when computing the hidden state vectors $\mathbf{h}_t$. Run the modified codes on the given input. Lastly report (10 pts) and make sense (15 pts) of your observations.

   [[[With the tanh activation function, the output is <==

   hidden state at time 0: [1. 1. 1.]
   hidden state at time 1: [0.99998771 0.99998771 0.99998771]
   hidden state at time 2: [0.99998771 0.99998771 0.99998771]
   hidden state at time 3: [0.99998771 0.99998771 0.99998771]
   hidden state at time 4: [0.99998771 0.99998771 0.99998771]

   The tanh function saturates to close to 1 quickly as the positive input increases. Without that, the output is

   hidden state at time 0: [1. 1. 1.]
   hidden state at time 1: [6. 6. 6.]
   hidden state at time 2: [21. 21. 21.]
   hidden state at time 3: [66. 66. 66.]
   hidden state at time 4: [201. 201. 201.]

   Since all matrices and vectors are taking value one, the addition in the linear function

   $$\mathbf{h}_t = +W\mathbf{h}_{t-1} + U\mathbf{x}_t \tag{1}$$

   can only grow the magnitudes of the elements in $\mathbf{h}_{t-1}$.]]]

2. Prove that the CFG grammar below allow derivations of infinite length and thus generate a language with sentences with infinite length (15 pts). Then justify or disapprove the existence of such derivations (10 pts).

   [[[The right hand side of the production *Nominal*→*Nominal Noun* contains *Nominal* that can be recursively re-written by the production itself, without eliminating the non-terminal *Nominal*. <==

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

Figure 1: A CFG

One possible answer: such recursion is needed as human languages are recursive so the production captures the humans languages patterns.

Other reasonable and well-justified answers will be accepted.]]]

3. In Lecture 16-17, we showed how the CYK algorithm fills out the last column of the parsing table on the sentence "*Book the flight through Houston*". Explain how the algorithm fills out the cell $(0, 1)$ and $(0, 3)$, using the CFG in CNF on the slides for Lecture 16.

   (*Hints: read the CYK algorithm section in SLP textbook and pay attention to title of the table that contains the converted CFG in CNF.*)

   [[[For the cell $(0, 1)$, the non-terminal S is the based on the production $S \rightarrow book|include|prefer$. <== $Nominal \rightarrow book|flight|meal|money$ and $VP \rightarrow book|include|prefer$ give $Nominal$ and VP. The original lexical entries from the CFG before the conversions carry over unchanged as well, so $Verb \rightarrow book$ and $Noun \rightarrow book$ give $Verb$ and $Noun$ in the cell.

   For the cell $(0, 3)$, S comes from the rule $S \rightarrow VerbNP$ with the breaking point $k = 1$ (i.e., pull $Verb$ from the cell $(0, 1)$ and NP from the cell $(1, 3)$. VP and $X2$ come from $VP \rightarrow VerbNP$ and $X2 \rightarrow VerbNP$, respectively, with the same breaking point.]]]

4. The following two figures show a PCFG and the table for calculating inside probabilities during probabilistic syntactic parsing. Explain why only β_{S} is shown in cell $(1,3)$ (10 pts) and how the probability is calculated using the inside algorithm (15 pts).

S → NP VP	1.0	NP → NP PP	0.4
PP → P NP	1.0	NP → *astronomers*	0.1
VP → V NP	0.7	NP → *ears*	0.18
VP → VP PP	0.3	NP → *saw*	0.04
P → *with*	1.0	NP → *stars*	0.18
V → *saw*	1.0	NP → *telescopes*	0.1

	1	2	3	4	5
1	$\beta_{\mathrm{NP}} =$ 0.1		$\beta_{\mathrm{S}} =$ 0.0126		$\beta_{\mathrm{S}} =$ 0.0015876
2		$\beta_{\mathrm{NP}} =$ 0.04 $\beta_{\mathrm{V}} =$ 1.0	$\beta_{\mathrm{VP}} =$ 0.126		$\beta_{\mathrm{VP}} =$ 0.015876
3			$\beta_{\mathrm{NP}} =$ 0.18		$\beta_{\mathrm{NP}} =$ 0.01296
4				$\beta_{\mathrm{P}} =$ 1.0	$\beta_{\mathrm{PP}} =$ 0.18
5					$\beta_{\mathrm{NP}} =$ 0.18
	astronomers	*saw*	*stars*	*with*	*ears*

Figure 2: Probabilistic CFG and Inside probability calculation.

[[[The cell $(1,3)$ includes all possible ways to break the sub-sentence from word 1 to 3 into two subtrees. The breaking points can be 1 and 2. If it is broken at 1, then cells (1,1) has an NP and cell (2,3) has a VP. According to the PCFG, $S \to$ NP VP is a valid derivation. The second breaking point 2 will not lead a valid parse since the cell (1,2) is empty (no parse for words 1 to 2). We can find an inside probability β_S in cell (1,3) by <==

$$\beta_S = \beta_{NP}\beta_{VP}\Pr(S \to NPVP) = 0.1 \times 0.126 \times 1 = 0.0126. \tag{2}$$

]]]

5. Use the above PCFG to find a parsing tree T for the sentence S="*telescopes saw stars*" (10 pts). Then calculate the probability of $\Pr(T)$ (15 pts).

[[[By applying the rules S→NP VP, NP→ telescopes, VP→ V NP, V→saw, and NP→ stars, we obtain a parsing tree for the sentence. The probability $\Pr(T)$ is the product of these 5 rules defined in the PCFG. $\Pr(T) = 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.18$.]]] <==