CS378 Spring 2020 Midterm Review

1 PCFGS/CKY

Here's a PCFG with start symbol VP, nonterminals {VP, PP, P, V, NNS}, and terminals {sells, to, books, meetings}.

VP → V NNS PP [0.5]
VP → V NNS [0.5]
PP → P NNS [1.0]

V → sells [0.5]
V → books [0.5]
P → to [1.0]
NNS → books [0.5]
NNS → meetings [0.5]

1. Apply lossless binarization to this grammar to obtain a binary grammar.
Replace the top rule with these two:
VP → V VP[NNS PP] [0.5]
VP[NNS PP] → NNS PP [1.0]
You can also binarize in the other direction or use different ways of introducing the intermediate symbol. The most important thing is to make sure your symbol contains all of the information about what still needs to be generated (this is what makes the binarization lossless) and that you still have a valid PCFG (rule probabilities sum to 1 for each symbol in the grammar).

2. Fill in the CKY chart for *sells books*. Use log base 2.
The chart has three cells with the following constituents and scores (all others have negative infinity log probability)

	VP: -3
V:-1	V: -1/NNS:-1

3. Fill in the CKY chart for *books meetings*. Use log base 2.
The chart has three cells with the following constituents and scores (all others have negative infinity log probability)

	VP: -3
V:-1/NNS:-1	NNS:-1

4. Fill in the CKY chart for *sells books to books*. Use log base 2.
Using the binarization above:

		VP: -4	
	NONE	VP[NNS PP]:-2	
VP:-2	NONE	PP:-1	
V:-1	V:-1/NNS:-1	P:0	V:-1/NNS:-1

Note that the VP[NNS PP] cell is populated using *books* | *to books* (split that way) and the top cell is populated used *sell* | *books to books*, taking a value from the bottom-left-most cell and the VP[NNS PP] cell. NONE is used to indicate cells for which all log probabilities are negative infinity.

2 Skip-gram

The skip-gram model is defined by

$$P(\text{context} = y|\text{word} = x) = \frac{\exp\left(\mathbf{v}_x \cdot \mathbf{c}_y\right)}{\sum_{y'} \exp\left(\mathbf{v}_x \cdot \mathbf{c}_{y'}\right)}$$

where x is the "main word", y is the "context word" being predicted, and $\mathbf{v}$, $\mathbf{c}$ are d-dimensional vectors corresponding to words and contexts, respectively. Note that each word has independent vectors for each of these, so each word really has two embeddings.

The skip-gram model considers the neighbors of a word to be words within a k-word window on either side (i.e., $k = 1$ gives the two immediately adjacent words). The skip-gram objective, log likelihood of this training data, is $\sum_{(x,y)} \log P(y|x)$, where the sum is over all training examples.

1. What happens to the number of training examples if $k = 5$?

   Increases by a factor of 5. A word is associated with *each* word sequentially in a window of size k. So if k=5, we get 10 training examples per word instead of 2 for $k = 1$.

2. How does the runtime change with larger k?

   The skip-gram computation itself has the same runtime, but training takes 5x longer if you run for the same number of epochs.

3. Think about the context of the word *balloons* in the following sentences:

he blew up balloons for the birthday party
I popped balloons using a pin because I didn't like seeing the bright colors

In these contexts (and more generally), what do you think will change about what the skip-gram model learns for *balloons* as you change $k = 1$ to $k = 3$? What about $k = 10$?

Nearby words like *up* and *for* highly indicate what syntactic context a word appears in; for example, it's often after a particle/preposition, after a verb, before a verb, etc. As k grows, this distribution is less indicative of the particular context.