Project 1

Kekoa Riggin Ling 473 August 3, 2017

Results

Constituent	PTB symbol	Count
Sentence	(S ...)	4671
Noun Phrase	(NP ...)	13221
Verb Phrase	(VP ...)	7920
Ditransitive Verb	(VP *verb* (NP ...) (NP ...))	48
Intransitive Verb	(VP *verb*)	123

My Approach to the Problem

I took a brute-force approach to counting the constituents in each of the Penn Treebank files.

First, I looped through each of the files in the annotated corpus directory and opened each file individually. I read the entire text one character at a time and checked for the `(` character. If the character in current index matched `(`, I checked the next four indexes for `S`, `VP`, and `NP` with any variation of a single space before or after. This allowed for minor inconsistencies related to whitespace would not affect my counting. This also prevented counting constituents like `NP-SUBJ`.I recognize that this is an inefficient method because any other inconsistencies will have affected my accuracy.

If the next four indexes contained a match to one of my constituents, I incremented the counter for the corresponding constituent for the file. Then, a list was added to a stack of the constituents. The list contained a string of the constituent symbol, an int for `NP` child nodes, and a int for any child nodes. The ints were used to determine whether a VP was also a ditransitive or intransitive.

If the character at the current index was not a `(`, then I checked if it was a `)`. If it was a `)`, then the last item in the stack was popped off. If that item was a `NP`, I checked the new last item to see if it was `VP`. If it was, then I incremented the first int in the list, which corresponds to the ditransitive count. If the new last item was `VP` regardless of the popped item type, I incremented the second int, which corresponded to intransitive count.

If that item was a `VP`, then I checked the corresponding ints. If those ints indicated that the VP

was either ditransitive or intransitive, then I incremented the corresponding counter for the file. Any `VP` that had a `NP` child node count of two was ditransitive and any `VP` that had a child node count of zero was intransitive. I see now that I should have also made the ditransitive counter ensure that the total child count was also two because, as it is, my counter could allow a ditransitive to have two `NP` child nodes **and** any combination of additional immediate child nodes.

When my counter had reached the last character in the file, the counts for the file were printed to a separated `.txt` for accuracy checking and then added to the universal counters that kept the total for all the files. Then the loop refreshed all of the local variables and went to the next file until the entire batch was complete.

Lastly, the total counts were printed to the console.

Reflection

1. I was proud of myself for discovering that a ditransitive verb would be in the last place of a stack 3 times (when it is initially pushed, after the first `NP` is popped off, and after the second `NP` is popped off). However, as explained above, I was disappointed that I did not account for non ditransitives that feature combinations of 2 two `NP` children and others.

2. I don't quite remember why, but I set up my stack to start with an empty value in index 0. I believe this has to do with the `( )` that surround the entire sentence in Penn Treebank notation. When faced with an empty file, my program throws an error. Unfortunately, I do not believe I have time to fix this.

3. If I had more time, I would spend more time on making my counter able to count constituents without checking the next four indexes for a match. Although I consider myself a novice programmer, I think I might have been able to solve this with more time.