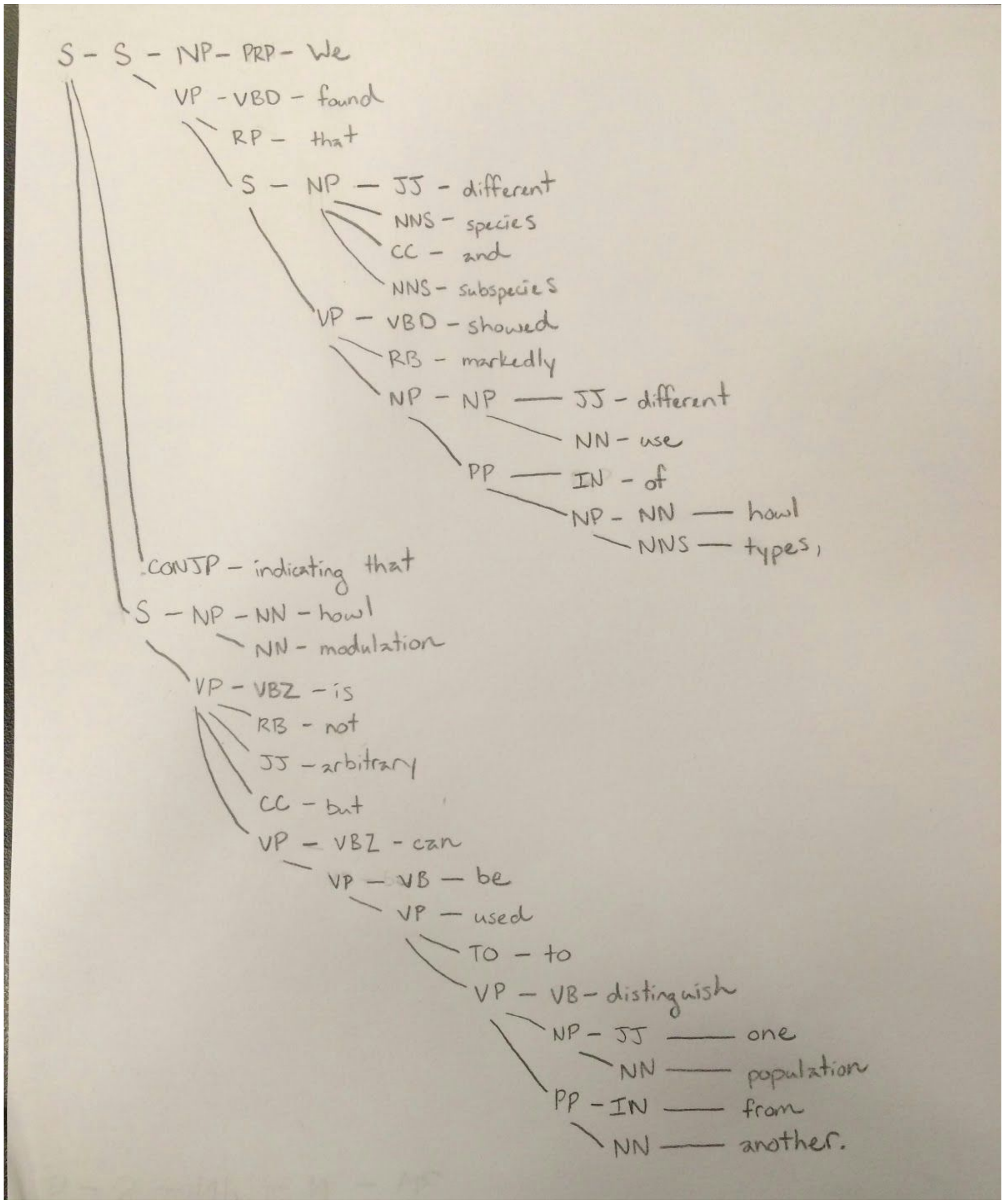
Katy Herrick
NLP #3 — March 18th, 2016

Manual Parse

Phrase Structure Tree:



Rules used:

S → S CONJP S	*CONJP → "indicating “that"*
S → NP VP	*PRP → "We"*
NP → PRP	*VBD → "found" | "showed"*
VP → VBD RP S	*RP → "that"*
NP → JJ NNS CC NNS	*JJ → "different" | "arbitrary," | "one"*
VP → VBD RB NP	*NNS →"species" | "subspecies" | "types,"*
NP → NP PP	*CC -→ "and" | "but"*
NP → JJ NN	*RB → "markedly" | "not"*
PP → IN NP	*NN → "use" | "howl" | "modulation" | "population" | "another."*
NP → NN NNS	*IN → "of" | "from"*
NP → NN NN	*VBZ → "is" | "can"*
VP → VBZ RP JJ CC VP	*VB → "used" | "distinguish" | "be"*
VP → VBZ VP	*TO → "to"*
VP → VB VP	
VP → VBZ TO VP	
VP → VB NP PP	
PP → IN NN	

Difficulties:

1. The sentence is long, so even finding a way to visualize and annotate it without making the result convoluted was challenging.
2. The rules of English' are too basic to handle the complex subordinating clauses found in this sentence.
    a. In particular, “indicating that” was a difficult phrase to handle, so I just made it its own thing. I realize a true **CONJP** would start with a coordinating conjunction and not the word “indicating.”
    b. The fact that the clause “but can be used” omits the subject was also difficult. I decided to put everything from “is not arbitrary” to the end of the sentence as one big verb phrase because they both need to be connected to the subject “howl modulation”!

Recursive Descent Parser

Grammar #1: I used the same grammar rules I used in my manual parse, and after a few moments waiting for the parser to run, I got a huge stack of error messages, terminating with these:

```
  File "/Users/Katy/cl/env/lib/python2.7/site-packages/nltk/tree.py", line 156, in __getitem__
    return self[index[0]]
  File "/Users/Katy/cl/env/lib/python2.7/site-packages/nltk/tree.py", line 151, in __getitem__
    return list.__getitem__(self, index)
RuntimeError: maximum recursion depth exceeded while calling a Python object
```

It seems that something in my rules is recursive, so the parser got confused and didn't know how to terminate.

Grammar #2: I realized my rules were a little complicated, so I did something silly and revised my rules as follows:

S -> NP VP
NP -> "We"
VP -> "found" "that" "different" "species" "and" "subspecies" "showed" "markedly" \
"different" "use" "of" "howl" "types," "indicating" "that" "howl" "modulation" \
"is" "not" "arbitrary," "but" "can" "be" "used" "to" "distinguish" "one" \
"population" "from" "another."

Not surprisingly, the parser was able to handle this quickly, although it came out exactly as you'd expect. (It's one huge column where each word comes out on a new line.)

Grammar #3: I tried to make a middle ground between grammars #1 and #2, but it didn't take much complexity to start throwing the parser into the infinite loop again. I finally decided on the following set of rules.

S -> NP VP NP VP NP | NP VP CC VP VP NP IN NP| S CONJP S
NP -> N | N CC N | ADVP NP | N IN NP | "howl" "types," | "howl" "modulation" | \
"one" "population" | "another." | "not" "arbitrary,"
VP -> V | "to" **V | V ADJP | V VP** | "found" "that" | "be" "used" |
CONJP -> "indicating" "that"
ADVP -> "markedly" "different"
N -> "We" | "different" "species" | "subspecies" | "use"
V -> "showed" | "distinguish" | "is" | "can"
CC -> "and" | "but"
IN -> "from" | "of"
ADJP -> "not" "arbitrary,"

Given those rules, it gave the following output.

```
(S
  (S
    (NP (N We))
    (VP found that)
    (NP (N different species) (CC and) (N subspecies))
    (VP (V showed))
    (NP
      (ADVP markedly different)
      (NP (N use) (IN of) (NP howl types,))))
  (CONJP indicating that)
  (S
    (NP howl modulation)
    (VP (V is) (ADJP not arbitrary,))
    (CC but)
    (VP (V can) (VP be used))
    (VP to (V distinguish))
    (NP one population)
    (IN from)
    (NP another.)))
```

Shift-Reduce Parser

Grammar #1: Knowing that it probably wouldn't work, I used the grammar I used in my manual parse again. As expected, when I ran the shift-reduce parser, it didn't output a tree. So I ran the parser again with the trace on, and it gave me a long trace showing that the parser had tried, ultimately ending up with the following:

```
R [ S PP NNS CONJP NP VBZ RB JJ CC VBZ VB VB TO VB NP * ]
```

It looks like some of the intermediary stages of my tree, but ultimately the parser wasn't able to resolve the input as a sentence given these grammar rules.

Grammar #2: As expected, the oversimplified grammar #2 from above has the same results as with the Recursive Descent Parser.

Grammar #3: Creating the parser with the grammar #3 from above throws many warnings about unused rules. The parser does not find a parse.

Grammar #4: I came up with the following set of rules, though they don't necessarily produce a syntactically correct parse.

> **S -> S VP NP | NP VP NP | S CONJP S**
> **NP** -> "We" | "different" "species" "and" "subspecies" | "markedly" \
> "different" "use" "of" "howl" "types," | "howl" "modulation" | \
> "one" "population" "from" "another."
> **VP -> VP CC VP** | "found" "that" | "showed" | \
> "is" "not" "arbitrary," | "can" "be" "used" "to" "distinguish"
> **CONJP ->** "indicating" "that"
> **CC -> "but"**

In particular, the rule **S -> NP VP NP** doesn't work because "`(NP We) (VP found that) (NP different species and subspecies)`" is not a complete sentence. Due to nltk's lack of backtracking in its ShiftReduce algorithm, this was the best I could do to get a tree as output at all!

```
(S
  (S
    (S (NP We) (VP found that) (NP different species and subspecies))
    (VP showed)
    (NP markedly different use of howl types,))
  (CONJP indicating that)
  (S
    (NP howl modulation)
    (VP
      (VP is not arbitrary,)
      (CC but)
      (VP can be used to distinguish))
    (NP one population from another.)))
```

Left-Corner Parser

Grammar #1: After running the original rules through a third parser, I realize they were too complex even for me to understand, and I am probably missing a rule or two that would allow the parser to terminate. However, as they stand now, the rules do not generate a tree.

Grammar #2: Same results as in the other parsers. Runs quickly, but not useful.

Grammar #3: NLTK was not able to create a tree out of these rules, but more interestingly, it couldn't even create the Left Corner Parser object! It throws the following warning

```
ValueError: LeftCornerParser only works for grammars without empty
productions.
```

Grammar #4: I modified grammar #4 from the Shift Reduce Parser experiment to get the rules as follows.

> **S -> NP VP NP | NP VP NP VP NP | S CONJP S**
> **NP -> N | NP CC N | ADJ N | ADVP NP | NP PP | N N**
> **VP -> V | V VP |VP CC VP | V ADJP** | "to" **V** | "found" "that"
> **CONJP ->** "indicating" "that"
> **PP -> P NP**
> **ADVP -> ADV ADJ**
> **ADJP -> ADJ** | "not" **ADJ**
> **ADV ->** "markedly"
> **ADJ ->** "arbitrary," | "different" | "one"
> **CC ->** "but" | "and"
> **N ->** "We" | "subspecies" | "use" | "another." | "howl" | "types," | "modulation" | "species" | "population"
> **V ->** "showed" | "is" | "can" | "be" | "used" | "distinguish"
> **P ->** "of" | "from"

I did my best to write rules here that actually match English, but I'm still unsure of the linguistic classification of certain phrases and especially of the word "that", which can take on many roles. In any case, the parser works quite well! It is very quick compared to the other two parsers.

```
(S
  (S
    (NP (N We))
    (VP found that)
    (NP (NP (ADJ different) (N species)) (CC and) (N subspecies))
    (VP (V showed))
    (NP
      (NP (ADVP (ADV markedly) (ADJ different)) (NP (N use)))
      (PP (P of) (NP (N howl) (N types,)))))
  (CONJP indicating that)
  (S
    (NP (N howl) (N modulation))
    (VP
      (VP (V is) (ADJP not (ADJ arbitrary,)))
      (CC but)
      (VP (V can) (VP (V be) (VP (V used) (VP to (V distinguish))))))
    (NP
      (NP (ADJ one) (N population))
      (PP (P from) (NP (N another.))))))
```