Alex Ginella

Ethan Poole

Comp Ling

May 11 2019

The Write up

I found that the most challenging part of creating the POS tagger was the tag function. I will primarily be talking about tag as it does the bulk of the work and pulls together all the components from the earlier parts of the project. My thinking for tag is as follows:

1) Break the input string up into an array and find all the variants of each word using the corpus:

```
taggedString = [concat $ map (\word -> [head $ findPOS word]) string]
```

2) My taggedString array is a sub component of the final output as it only contains only one variant of the input string str. I will re-concatinate other variants onto this later
3) I keep track of all of the pos variants of all of the words in str in an array called partsOfSpeech:

```
partsOfSpeech = concat $ map (\word -> findPOS word) (words str)
```

4) Now the tricky part. I have to modify **taggedString** to encorporate all the variants of each word in the string contained in **partsOfSpeech**. I traversed both arrays and reconcatenated **taggedString** back onto itself changing the pos of one of the words each time until I'd fully traversed **partsOfSpeech**

```
sentences = nub $ concat $ map (\(TaggedWord (word, pos)) ->
taggedString ++ [map (\(TaggedWord (wrd, ps)) -> if (wrd == word)
then (changeTag (TaggedWord (wrd, ps)) pos) else ((TaggedWord (wrd,
ps)))) (concat taggedString)]) partsOfSpeech --ouch
```

(i wont be offended if you wince)

5) **sentences** is essentially the final output but without the probabilities, so now I add those in. I do this by using some of the functions I'd written previously, such as posProbSLG and valP. For each sentence string in sentences I call valP to calculate the probability of the string and then I attatch it to the sentence in the

form of a tuple:

```
output = map (\string -> (string, probabilityOfString string))
sentences
probabilityOfString string = valP probSLG (posString string)
probSLG = posProbSLG corpus
```

And thats pretty much it. I gotta say, I dont think threre's much I would change, obviously I didnt make use of the sanitize function, but I was happy enough with my output and accuracy to leave it be for now. Something my program doesn't do that would be cool is check to see if a word is represented inside the corpus or not. If you input a string of words in which one or more of them are not in the corpus the program will fail. I also noticed that my tag function doesn't quite find ALL the possible combinations of sentences, for example `tag corpus5 "the cat"` only outputs three distinct sentences instead of four.