Yuankun Zhu (jennyykz)
EECS595: Natural Language Processing
Homework 3 Assignment

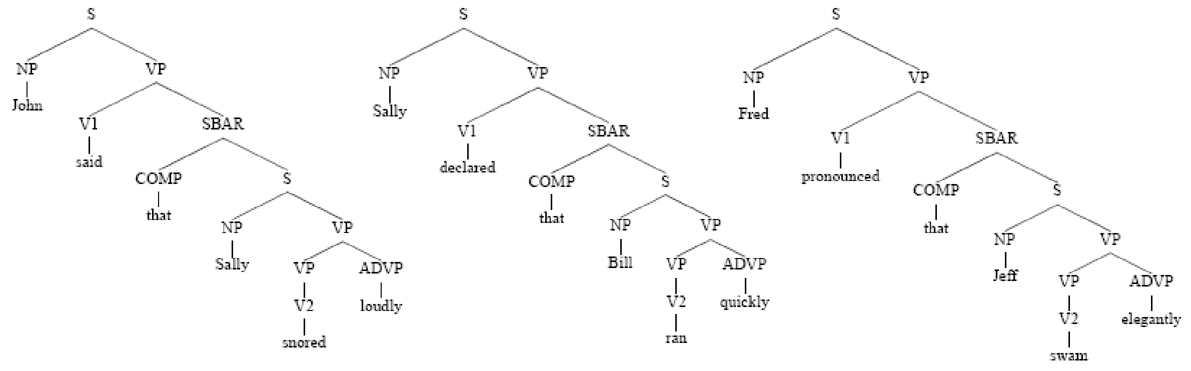
November 4, 2019

Due date and time: 11:59pm, November 3, 2019

This assignment includes a written component and a programming component.

1 Written Assignment

1. Your friend decides to build a Treebank. He finally produces a corpus which contains the following three parse trees:

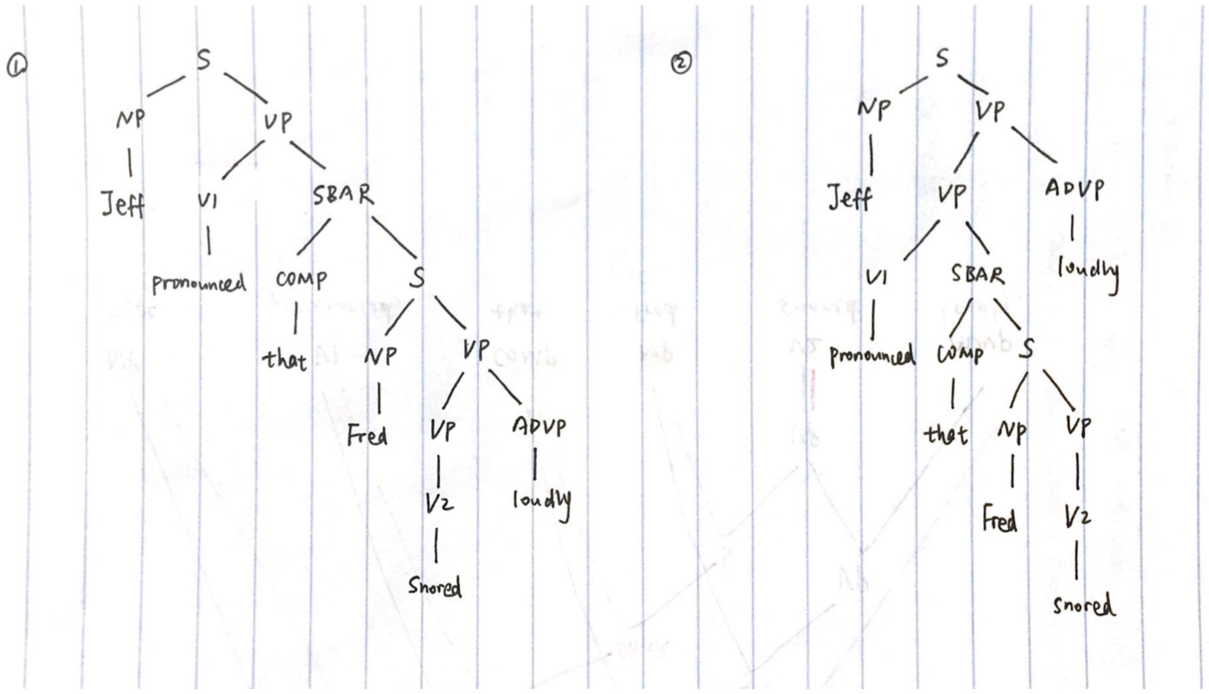


   You then purchase the Treebank and decide to build a PCFG, and a parser, using your friend's data. Now answer the following three questions:

   - Show the PCFG that you would derive from this Treebank.

Rules			P
S	→	NP VP	1.0
SBAR	→	COMP S	1.0
COMP	→	that	1.0
ADVP	→	loudly | quickly | elegantly	0.33
VP	→	V1 SBAR	0.33
VP	→	VP ADVP	0.33
VP	→	V2	0.33
V1	→	said | declared | pronounced	0.33
V2	→	snored | ran | swam	0.33
NP	→	John | Bill | Fred | Jeff	0.17
NP	→	Sally	0.33

- Show two parse trees for the string"Jeff pronounced that Fred snored loudly", and calculate their probabilities under the PCFG.



For the first parse tree, the probability is:

$$1 \times \frac{1}{6} \times \frac{1}{3} \times \frac{1}{3} \times 1 \times 1 \times 1 \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{6} = \frac{1}{26244}.$$

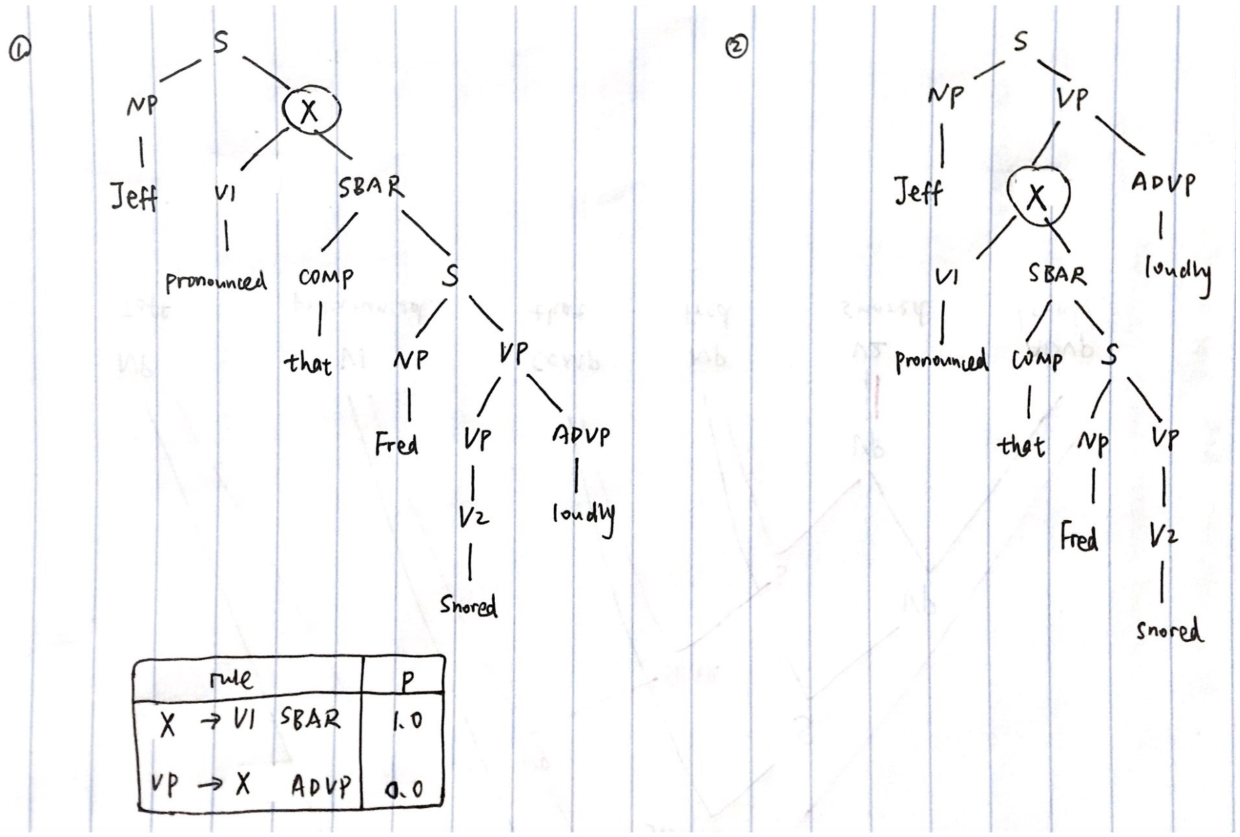
For the second parse tree, the probability is:

$$1 \times \frac{1}{6} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} \times 1 \times 1 \times 1 \times \frac{1}{6} \times \frac{1}{3} \times \frac{1}{3} = \frac{1}{26244}.$$

The probabilities of the two parse trees under the PCFG are the same.

- You are shocked and dismayed, that “Jeff pronounced that Fred snored loudly” has two possible parses, and that one of them-that Jeff is doing the pronouncing loudly-has relatively high probability, in spite of it having the ADVP “loudly” modifying the *higher* verb, “pronounced”. This type of high attachment is never seen in the corpus, so the PCFG is clearly missing something. You decide to fix the Treebank, by altering some non-terminal labels in the corpus. Show one such transformation which results in a PCFG that gives zero probability to parse trees with *high* attachments. (Your solution should systematically refine some non-terminals in the Treebank, in a way that slightly increases the number of non-terminals in the grammar, but allows the grammar to capture the distinction between high and low attachment to VPs.)

  The modifications on PCFG are shown below:



rule	P
X → VI SBAR	1.0
VP → X ADVP	0.0

  In this way, the probability of the right parse tree (which corresponds to high attachments of ADVP) is 0 while the the probability of the left parse tree (which corresponds to low attachments of ADVP) is greater than 0.

2. Suppose we have the following portion of a mini-grammar in CNF:

   Given this grammar, use the probabilistic CKY algorithm to parse the sentence *“Book the flight with the money”*. You should clearly present the filled chart by the CKY algorithm (as shown in the example discussed in the lecture notes).

Rules			P
S	$\rightarrow$	NP VP	0.8
S	$\rightarrow$	VP PP	0.05
NP	$\rightarrow$	Det N	0.3
VP	$\rightarrow$	V NP	0.2
PP	$\rightarrow$	Prep NP	1.0
V	$\rightarrow$	includes	0.05
V	$\rightarrow$	book	0.30
Det	$\rightarrow$	the	0.4
Det	$\rightarrow$	a	0.4
Prep	$\rightarrow$	with	0.5
N	$\rightarrow$	meal	0.01
N	$\rightarrow$	flight	0.02
N	$\rightarrow$	money	0.03

Book	the	flight	with	the	money.
V $\rightarrow$ book 0.30		VP $\rightarrow$ V NP $0.2 \times 0.3 \times 0.0024$ $= 1.44 \times 10^{-4}$			S $\rightarrow$ VP PP 1.296×10^{-8}
	Det $\rightarrow$ the 0.4	NP $\rightarrow$ Det N $0.008 \times 0.3 = 0.0024$	PP $\rightarrow$ NP Prep $1 \times 0.0024 \times 0.5$ $= 0.0012$		
		N $\rightarrow$ flight 0.02			
			Prep $\rightarrow$ with 0.5		PP $\rightarrow$ NP Prep 1.8×10^{-3}
				Det $\rightarrow$ the 0.4	NP $\rightarrow$ Det N 0.012×0.3 $= 0.0036$
					N $\rightarrow$ money 0.03

1.1 What to Submit?

You will need to upload a PDF file titled “HW3-written.pdf” to the CANVAS system. For those who are familiar with Latex, the latex files for the assignment will be made available. You can directly add your answers to each of the questions. But this is not a requirement.

2 Programming Assignment

In this programming assignment you will implement a parser based on the CKY Algorithm using any programming language of your choice. Note the algorithm described in the class (in the lecture note) is a recognizer not a parser. Make the changes necessary to turn it into a parser.

2.1 Implementation Details

Your parser should read a file containing a list of sentences to parse, one sentence per line. The output should consist of labeled bracketed representations of all the parse trees that correspond to the input sentences, or an indication that the sentence failed to parse (i.e., output "FAIL"). For example, the following would be the appropriate parse for the sentence "take the green block":

[S [Verb take] [NP [Det the] [Adj green] [Noun block]]]

Your program should be able to handle the following command line:

```
$ Parse TestInputFile GrammarFile TextOutputFile
```

Where

- `TextInputFile` is a set of sentences to be parsed, one line for each sentence.
- `GrammarFile` is a file with grammar rules in the form of context free grammar (see below).
- `TextOutputFile` is a file that contains the parsed results in the bracketed representations as shown in the example above. One line for each parsed result.

2.2 Grammar

You will need to first come up with a set of grammar rules based on the training examples. You can either manually create the grammar rules or write a program to automatically extract grammar rules. In the latter case, you don't need to submit your program. All we need is the file with grammar rules (i.e., `GrammarFile` mentioned above). Your parser should first load in the grammar rules and convert them to Chomsky Normal Forms. After you get the parsing results, you will also need to convert the structure (binary tree) back to the tree in the original form (as shown in the example).

The following examples files are provided in the assignment directory for you to identify the grammar rules.

- Training sentences (raw data): TrainingRaw.txt
- Training sentences with parsed results from which you can create grammars: TrainingTree.txt
- Vocabulary: Vocabulary.txt

2.3 Testing

After implementing the algorithm, you can test whether your parser can correctly parse the sentences using the following files which are available in the assignment directory:

- Testing sentences (raw data): TestingRaw.txt
- Testing sentences with parsed results to evaluate your parser: TestingTree.txt

2.4 What to Submit?

You will need to submit the following files:

- Your source code
- The grammar file (context free grammar)
- Simple README file about how to compile and run your source code.

You need to make sure your code will compile and run on a CAEN machine. We will test your code on a separate set of testing sentences including ungrammatical ones.