Homework Assignment 2

Anand Kumar Singh (11392045)

November 22, 2017

1 Problem 1: CNF

The Chomsky Normal Form (CNF) of the given grammar is as follows:

- S → NP VP
- S → X Y
- X → *I*
- Y → VP PP
- NP → Det N
- VP → V NP
- V → *ate*
- PP → Pre NP
- Det → *the* | *a*
- N → *fork* | *salad*
- Pre → *with*

2 Problem 2: CYK

2.1 CYK Parsing Table

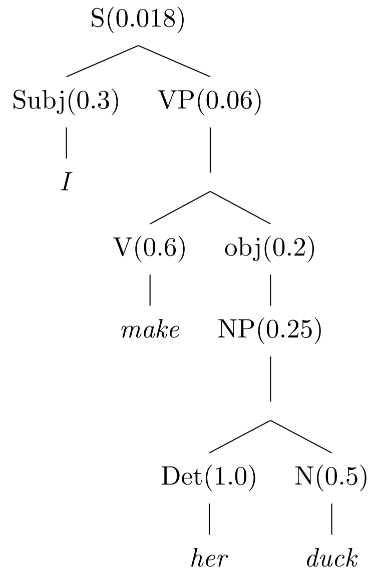
B:
VP → V obj (0.0600)

VP → V Small (0.0096)

A:
S → Subj VP (0.0180)

For a small grammar like the one given the CYK algorithm is feasible since it's $\mathcal{O}(n^3)$parsing in the length of the sentence (and the number of non-terminals in grammar) isn't too large. However for large grammars we need binarization/CNF form of the grammar to make CYK algorithm a feasible parser for them.

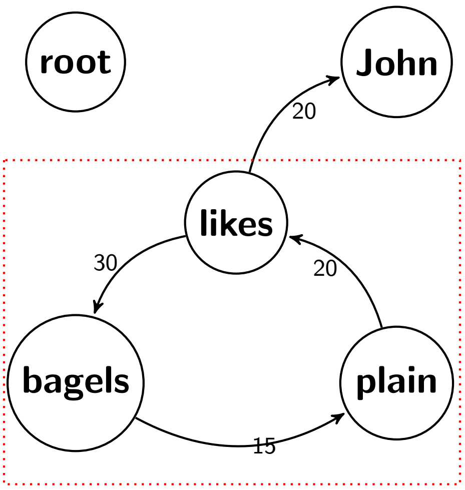
2.2 Most Probable Parse



3 Problem 3: Dependency Parsing / MST

3.1 CLE First Step

In the fist step of Chu-Li-Edmonds(CLE) every node greedily accepts the incoming edge with highest weight. This results in the following graph:



The cycle is indicated by the red dotted box.

3.2 CLE Last Step

Contracting the cycle to a single node and recalculating scores of incoming and outgoing edges:

Incoming arc weights:

$root \rightarrow likes \rightarrow bagels \rightarrow plain = 60$
$root \rightarrow likes \rightarrow plain \rightarrow bagels = 25$
$root \rightarrow bagels \rightarrow plain \rightarrow likes = 35$
$root \rightarrow bagels \rightarrow likes \rightarrow plain = 15$

$root \rightarrow plain \rightarrow likes \rightarrow bagels = 50$
$root \rightarrow plain \rightarrow bagels \rightarrow likes = 15$

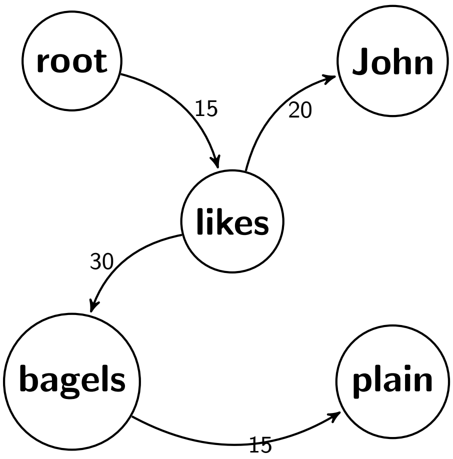
We therefore select $root \rightarrow likes \rightarrow bagels \rightarrow plain$

Outgoing arc weights:

$likes \rightarrow John = 20 > plain \rightarrow John = 10 > bagels \rightarrow John = 5$

We therefore select $likes \rightarrow John$

The resulting final graph is as follows:



4 Problem 4: Dependency parsing / Transition Based

Table 1: Arc-Standard Transition Based Dependency Parser

Transition	Stack	Buffer	Arcs	
	[ROOT]	[A koala eats leafs and barks]		
SHIFT	[ROOT A]	[koala eats barks and leafs]		
SHIFT	[ROOT A koala]	[eats leafs and barks]		
LEFT-ARC (det)	[ROOT koala]	[eats leafs and barks]	A ∪ det(koala,A)	
SHIFT	[ROOT koala eats]	[leafs and barks]		
LEFT-ARC (nsubj)	[ROOT eats]	[leafs and barks]	A ∪ nsubj(eats,koala)	
SHIFT	[ROOT eats leafs]	[and barks]		
RIGHT-ARC (dobj)	[ROOT eats]	[and barks]	A ∪ dobj(eats,leafs)	
SHIFT	[ROOT eats and]	[barks]		
RIGHT-ARC (cc)	[ROOT eats]	[barks]	A ∪ cc(eats,and)	
SHIFT	[ROOT eats barks]	[]		
RIGHT-ARC (conj)	[ROOT eats]	[]	A ∪ conj(eats,barks)	
RIGHT-ARC (root)	[ROOT]	[]	A ∪ root(root,eats)	

The arc standard transition based dependency parser in table 1 won't be able to correctly predict the given structure. The dependency arcs *cc (eats, and)* and *conj (eats,barks)* are wrongly predicted. The correct dependencies should be *cc (leafs, and)* and *conj (leafs,barks).* However since the transitions are not constrained to follow a particular order and are applicable as long as their preconditions are met, the arc standard transition parser that correctly predicts the given structure is presented in table 2.

Table 2: Corrected Arc-Standard Transition Based Dependency Parser

Transition	**Stack**	**Buffer**	**Arcs**	
	[ROOT]	[A koala eats leafs and barks]		
SHIFT	[ROOT A]	[koala eats barks and leafs]		
SHIFT	[ROOT A koala]	[eats leafs and barks]		
LEFT-ARC (det)	[ROOT koala]	[eats leafs and barks]	A ∪ det(koala,A)	
SHIFT	[ROOT koala eats]	[leafs and barks]		
LEFT-ARC (nsubj)	[ROOT eats]	[leafs and barks]	A ∪ nsubj(eats,koala)	
SHIFT	[ROOT eats leafs]	[and barks]		
SHIFT	[ROOT eats leafs and]	[barks]		
RIGHT-ARC (cc)	[ROOT eats leafs]	[barks]	A ∪ cc(leafs,and)	
SHIFT	[ROOT eats leafs barks]	[]		
RIGHT-ARC (conj)	[ROOT eat leafs]	[]	A ∪ conj(leafs,barks)	
RIGHT-ARC (dobj)	[ROOT eats]	[]	A ∪ dobj(eats,leafs)	
RIGHT-ARC (root)	[ROOT]	[]	A ∪ root(root,eats)	