ADVANCED METHODS IN NLP
EXERCISE #x SOLUTION

Uri Avron [uriavron@gmail.com] [308046994]
Ofri Kleinfeld [ofrik@mail.tau.ac.il] [302893680]
Ido Calman [calman.ido@gmail.com] [308353499]

June 9, 2018

Question 1

(a) The program is likely to create long sentences because of the rule [**NP -> NP PP**]. This is a **recursive** rule that may in the "best" case generate 5 terminals: Left hand side NP would derive [Det Noun] and right hand side PP would derive [NP Prep] then the latter NP would again derive [Det Noun]. In this case we would end up with: [Det Noun Prep Det Noun]. But each time we encounter an NP we have a **0.5 probability** to choose the derivation [**NP -> NP PP**] instead, and that would yield another "best case 5" addition.

(b) We may calculate the probability of a double adjective sentence. For such sentence to be generated, our grammar should choose twice consecutively the rule: [**Noun -> Adj Noun**] because then we would end up with [Noun -> Adj Adj Noun] as expected. Note that there are 5 other rules of the form [Noun -> *] which are all deriving terminals. So the probability when encountering a Noun to choose twice the specified rule is $\frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$ which is fairly rare.

(c) In order to generate shorter sentences and make it less rare to generate a double adjective sentence, we should change the weights of the rules: [**NP -> Det Noun**] from 1 to 2 and [**Noun -> Adj Noun**] from 1 to 5. This way the probabilty to derive a recursive NP is $\frac{1}{3}$ and the probability for a double adjective is $\frac{5}{10} = \frac{1}{2}$. Below are the results when running the generator 5 times:

a **perplexed delicious** chief of staff understood the sandwich .

is it true that a **fine perplexed perplexed fine** sandwich understood a pickle ?

is it true that a **delicious perplexed pickled** sandwich under the perplexed floor ate the **fine delicious** sandwich under a pickle ?

a sandwich with a president kissed a **perplexed perplexed** pickle .

every chief of staff under every sandwich wanted the pickle !

Question 3

(a) **General note**: more information appears in the grammar text file in comments. We go over the main rules that make our grammar work with the provided examples:

(a) "Sally" is a proper noun (an entity) thus [**S -> ProperNoun VP**] was added. Proper Nouns are different than Nouns because "the Sally", for example, is incorrect.

(b) Conjunctions (Conj) like "and", "or" were added and accordingly rules to

connect two NP / VP with a conjunction word in between.

(c) Intransitive verbs (IVerb) are verbs that appear without an object afterwards.

(d) Complementizer (Comp) was added for VP in the following way: [**VP -> Verb Comp S**] so a full sentence S can be expressed after the verb and the word "that"

(e) Added Pronouns (Pron) like "it" and rules that allow the sentences of the form: "it ... that"

(f) The word "very" is normally an adverb but for our case this is a quantifier for adjectives, so a recursive rule [**Adj -> Very Adj**] was added to support multiple "very"s

(g) Added [**VP -> Verb Prep NP**] so a verb can be performed on an NP with a preposition word (with, on, in, under) in between.

(h) Added [**VP -> Be Adj**] to express phrases like "Sally is lazy"

(i) Added (partial) support for Present Progressive with the rule [**VP -> Be PVerb NP**] where PVerb is the verb in the PP form (eating).

(j) The rule [**VP -> Be NP**] expresses phrases like "Sally is a sandwich"

(b) **Relative Clauses** have 2 different relations - The first is where the verb is relative: "... the president that **ate** the sandwich" so a rule [**VP -> Verb NP Clause Comp VP**] was added. The second is where the noun is the relative: "the sandwich that **the president** ate"so a rule [**VP -> Verb NP Clause Comp NP**] was added. Moreover, we allow some recursion to have consecutive "that"s with a special non-terminal "NPV" which derives NP VP. Then VP may again derive a relative clause and so on.

Singular vs Plural was implemented with special two non-terminals: NPS and VPS. We had to make another basic rule [**S -> NPS VPS**] so we decide in the beginning of the derivation if the whole sentence is plural or singular. This way we know that no matter what NPS we derive, the verb would still be in plural form. This assures that we do **not** derive sentences like "Sally choose the president" but we do support "Sally and the citizens choose the president".

(c) Below are some examples of CKY parsing:

Sally ate a sandwich .

(ROOT (S (ProperNoun Sally) (VP (Verb ate) (NP (Det a) (Noun sandwich)))) (Dot .))

the citizens choose the president .

(ROOT (S (NPS (Dets the) (Nouns citizens)) (VPS (Verbs choose) (NP (Det the) (Noun president)))) (Dot .))

Sally is eating a sandwich .

(ROOT (S (ProperNoun Sally) (VP (Be is) (PVerbNP (PVerb eating) (NP (Det a) (Noun sandwich))))) (Dot .))

is it true that Sally thought that the chief of staff and the citizens choose the president ?

(ROOT (IsItTrueThatS (Is is) (ItTrueThatS (It it) (TrueThatS (True true) (ThatS (That that) (S (ProperNoun Sally) (VP (Verb thought) (Comp_S (Comp that) (S (NPS (NP (Det the) (Noun (Chief chief) (OfStaff (Of of) (Staff staff)))) (ConjNP (Conj and) (NPS (Dets the) (Nouns citizens)))) (VPS (Verbs choose) (NP (Det the) (Noun president))))))))))) (Question ?))