RUBEN WOLHANDLER
LIAT SCHIFF

Assignment 2 NLP

Part 1:

1. Why does the program generate so many long sentences? Specifically, what grammar rule is responsible for that and why? What is special about this rule? Discuss.

   The program generates long sentences because there are recursive rules.
   The recursive rules are:
   -1 NP NP PP
   -1 PP Prep NP
   Indeed NP can derive NP PP and the new NP can derive NP PP again and again. As well PP can also derive Prep NP and NP derive NP PP and we can enter a very big loop (theoretically the loop can be infinite) and the program will generate very long sentences.

2. The grammar allows multiple adjectives, as in: “the fine perplexed pickle”. Why do the generated sentences show this so rarely? discuss.

   The only rule that allows multiple adjectives is the recursive rule: Noun Adj Noun.
   But for this rule to be applied two times in a row, the probability is very small. There are 6 rules for Noun so to choose these rules a second time we have a probability equal to ⅙. So it will be rare.
   (and to have more than two adjacent adjectives it will be less probable)

3. Which numbers must you modify to fix the problems in (1) and (2), making the sentences shorter and the adjectives more frequent?

   To fix the generative long sentences we give to the non-recursive rule **NP Det Noun** more weight I will give him 4 instead of 1. So when generating NP, we will choose only ⅕ the recursive rule **NP NP PP** and ⅘ the non-recursive rule. So we will get less probability to enter in a loop.
   To make the adjective more frequent we increase the weight of the rule: **Noun Adj Noun**.
   We choose a weight of 5 and we will get a ratio of 5:1:1:1:1:1. So when generating Noun we will choose ½ of the time the recursive rule Noun Adj Noun and we will generate multiple adjectives more frequently.

4. What other numeric adjustments can you make to the grammar in order to favor a set of more natural sentences? Experiment and discuss.

   We put more weight on the more common words like: ‘wanted’, ‘understood’, ‘president’, ‘sandwich’, and ‘fine’.
   We can also put more weight on the sentences ending with ‘.’ than sentences ending with ‘?’ or ‘!’.
   Common words will appear more often than rarer words, and there will be more sentences without question marks or exclamation points. This in our opinion will make the grammar more natural.

Part 2:

Rules that are used for sentence
(a) Sally ate a sandwich .

ROOT -> S .
S -> S_VBD # It was added to notice that the sentence is in the past form
S_VBD -> NP VP_VBD # “Sally ate a sandwich”
NP -> Nnp # We need this rule always when the word “Sally” is written.
Nnp -> Sally
VP_VBD -> Vbd NP # Past form verb followed by NP - “ate a sandwich”
Vbd -> ate
NP -> Det Noun # Determiner followed by Noun - “a sandwich”
Det -> a
Noun -> sandwich

Rules that are used for sentence
(b) Sally and the president wanted and ate a sandwich

ROOT -> S .
S -> S_VBD # The sentence is in the past form
S_VBD -> NP VP_VBD # “Sally and the president wanted and ate a sandwich ”
NP -> NP Cconj NP # Used for NP conjunctions phrases “Sally and the president”
NP -> Nnp # We need this rule always when the word “Sally” is written.
Nnp -> Sally
Cconj -> and
NP -> Det Noun # Determiner followed by Noun - “the president”
Det -> the
Noun -> president
VP_VBD -> Vbd NP # Past form verb followed by NP - “wanted and ate a sandwich”
Vbd -> Vbd Cconj Vbd # Used for Vbd conjunctions phrases - “ wanted and ate”
Vbd -> wanted
Cconj -> and
Vbd -> ate
NP -> Det Noun # Determiner followed by Noun - “a sandwich”
Det -> a
Noun -> sandwich

Rules that are used for sentence
(c) the president sighed .

ROOT -> S .
S -> S_VBD # The sentence is in the past form
S_VBD -> NP VP_VBD
NP -> Det Noun # Determiner followed by Noun - “the president”
Det -> the
Noun -> president
VP_VBD -> Vbd_intran # Past form verbs that are intransitive - there are verbs that don't need to be followed by an object.
Vbd_intran -> sighed

Examples of other sentences with an intransitive verb are:” The president worked” or “Sally danced”, In the last sentence, *Sally* is the subject, and *dance* is the intransitive verb. There is not and cannot be a direct object that follows the sentence. Yet, the sentence can precede a preposition phrase or an adverb.

Note: to simplify, for the rest, We will only explain the new and the main rules that We haven't mentioned so far.

(d) the president thought that a sandwich sighed .

VP_VBD -> Vbd_sconj SBAR # "thought that a sandwich sighed ."
SBAR -> Sconj S_VBD # Subordinate clause: that + past sentence - “that a sandwich sighed"
Sconj -> that # subordinating conjunction (a subordinating conjunction is a conjunction that links constructions by making one of them a constituent of the other.
Vbd_sconj -> thought

This sentence presents an interesting phenomenon of some verbs connected with reporting can be followed by a that-clause acting as the direct object.

(e) it perplexed the president that a sandwich ate Sally .

S_VBD -> Prp Vbd_np_sconj NP SBAR # "it perplexed the president that a sandwich ate Sally ."
Prp -> it # To be able to start a sentence with personal pronoun
Vbd -> np_sconj perplexed

This sentence presents another interesting phenomenon of past form verbs that are followed by an indirect object and a that-clause.

(f) the very very very perplexed president ate a sandwich .

NP -> Det ADJP Noun # "the very very very perplexed president"
ADJP -> ADVP Adj # "the very very very perplexed"
ADVP -> Rb ADVP # "very very very"
ADVP -> Rb
Rb -> very # Adverb
ADJP -> Adj # "perplexed"

These new rules will let us create sentences with adverb phrases and adjective phrases.

(g) the president worked on every proposal on the desk .

VP_VBD -> Vbd_prep PP # "worked on every proposal on the desk"
Vbd_prep -> worked
PP -> Prep NP # "on every proposal on the desk"
Prep -> on
NP -> Det Noun PP # "every proposal on the desk"

This sentence presents a phenomenon of past form verbs and nouns that are followed by prepositions.

(h) Sally is lazy .

S -> S_PRESENT # For generating sentences in the present form.
S_PRESENT -> NP VP_PRESENT # "Sally is lazy"
VP_PRESENT -> Vbz ADJP # "is lazy"
Vbz -> is

We added these rules for representing sentences in the present tense in the form of a noun phrase followed by "is" and an adjective phrase after it.

(i) Sally is eating a sandwich .

S -> S_PRESENT # For generating sentences in the present form.
S_PRESENT -> NP VP_PRESENT # "Sally is eating a sandwich"
VP_PRESENT -> Vbz Vbg NP # "is eating a sandwich"
Vbz -> is
Vbg -> eating # Verb gerund or present participle

We added these rules for representing sentences in the present progressive tense in the form of a noun phrase followed by
"is" + a verb gerund and a noun phrase.

(j) the president thought that sally is a sandwich .
SBAR Sconj NP Vbz NP # “that sally is a sandwich“

We added this rule for generating sentence (j) that contains a that-clause in the present form.

Question & Answer:
Furthermore, note that handling sentences (b) and (h)/(i) can interact in a bad way, to create ungrammatical sentences. You do not need to solve this issue in this part of the assignment, but you do need to discuss it and explain what the problem is, using an example and a short explanation.

> For creating sentence (b) we added the rule NP -> NP Cconj NP
> This rule allows concatenation of noun phrases by the word ‘and’ without limit (Recursion). This rule can conflict with the added rules for generating sentences (i) and (h) and ungrammatical sentences might be generated such as “Sally and the perplexed president ***is*** eating”, instead of, “Sally and the perplexed president ***are*** eating”
> or “Sally and the president ***is*** lazy” instead of, “Sally and the president ***are*** lazy”.

Part 4:

The two phenomena we choose to implement are (a): “a” vs “an” and (b) Yes-no questions.

- “a” vs “an”:
  “a” and “an” appear either before a noun or before an adjective (sometimes words can appear between the two).
  To handle this phenomenon we separate the nouns that start with vowels and the nouns that start with consonants, we call them VowNoun and Noun respectively.
  As well, we separate the adjectives in the same way. We call **Adj** the adjectives that start with consonants and **VowAdj** adjectives that start with vowels.
  We have made sure that if “a” is generated the **first** word that follows it will begin with a consonant, and if “an” is generated the **first** word that follows it begins with a vowel.
  We also removed the rule: **Det – > a,** to be able to deal specifically with this phenomenon.
  We have added the following rules (taken from the rule that starts with Det):
  NP→ a Noun # Noun is a noun that starts with a consonant
  NP → a Noun PP
  NP → an VowNoun # VowNoun is a noun that starts with a vowel
  NP → an VowNoun PP

  Also for each rule, each adjective or noun that doesn’t follow “a” or “an”, we add the same rule by replacing **Adj by VowAdj** or **Noun** by **VowNoun**. For example from this rule **NP→Det ADJP Noun** we added the following rules (because **Det** is different from “a” and “an”):
  NP → Det ADJP VowNoun
  NP → Det VowAdj Noun
  NP → Det VowAdj VowNoun

  Note: We use **VowAdj** instead of **VowADJP** because **ADJP** derives **Adj** or something that begins with **Rb**, but in our grammar, **Rb** always begins with a consonant. So to avoid writing unnecessary rules, we don’t use **VowADVP**.

  Other rules we have added for example:
  Noun → Adj Noun
  Noun → Adj VowNoun
  VowNoun →VowAdj Noun
  VowNoun→ VowAdj VowNoun

Explanation: The first terminal of the RHS needs to begin with a vowel if the LHS is in the form of a VowNoun.
The first terminal of the RHS needs to begin with a consonant if the LHS is in the form of a Noun.
But the other terminals of the RHS can either start with vowels or not as long as the terminal before it is different from "a" or "an".
(This logic is also applicable to other rules)
To manage this phenomenon: "a" vs "an", we used this type of logic and we added a multitude of rules (too exhaustive to present them all here), you can see them on the file grammar4.

- Yes-no question

To handle this phenomenon we have added the following rules:

ROOT → YNQ ?
YNQ → AUX NP Vp_intran # can sally work ?
YNQ → AUX NP Vp_trans NP # does sally eat an apple ?
YNQ → AUX NP Vp_sconj SBAR # does the president think that Sally worked ?
YNQ → AUX NP Vp_prep PP # does sally work with the president ?
YNQ → AUX Prp Vp_np_sconj NP SBAR # did it perplex the president that Sally ate an apple ?

YNQ → Vbz NP Vbg NP # is Sally eating an apple ?
YNQ → Vbz NP NP # is Sally an orange ?
YNQ → Vbz NP ADJP # is Sally delicious ?
YNQ → Vbz NP ADJP Cconj ADJP # is Sally delicious and lazy ?
YNQ → Vbz NP VowAdj # is Sally intelligent ?
YNQ → Vbz NP VowAdj Cconj ADJP # is Sally intelligent and delicious ?
YNQ → Vbz NP ADJP Cconj VowAdj # is Sally delicious and intelligent ?
YNQ → Vbz NP VowAdj Cconj VowAdj # is Sally actual and intelligent ?

AUX = auxiliary
Vbz = is
Vp_intran = Intransitive verb present form
Vp_trans = Transitive verb present form

Yes no question begins with auxiliary, we choose to use: does, did, will, can, is.
We also choose to differentiate the auxiliary "is" because the form of the question that starts with "is" is a little bit different. For example when the auxiliary is "is" we can have a question without a verb (without considering "is") But with other auxiliary Yes-No questions must have a verb in the question.

Also with the auxiliary “is” if there is a verb linked to the auxiliary “is”, it will be on the -ing form, unlike the other auxiliary.
Generally, for the other auxiliaries, the form of these questions begins with the auxiliary followed by a subject NP (transitive or intransitive) and followed by a verb phrase(s) or an adjective phrase(s).

Example of yes-no question generated:

- *is the actual desk eating the apple in a very actual chief of staff ?*
- *did an authority work in the floor ?*
- *is the desk eating the intelligent hour ?*
- *is an orange important and very very delicious ?*
- *does the desk want an apple ?*
- *will a president sigh ?*
- *is an old old president doing an orange ?*
- *is a president an important pickle ?*
- *can the artist eat on the floor ?*
- *will a very perplexed lazy lazy floor think that it perplexed the pickle that the pickle wanted a desk ?*

Part 5:

We have extended our grammar by the following main phenomena:

- Plural VS Singular.
- Sentences in the present perfect form.
- Sentences in the form: has\have + NP\NNS(plural-noun)
- Sentences in the form: ‘has\have to’ + base-form verb + NP\NNS
- Sentences in the form: should/could + have + past-form verb

Plural VS Singular

We extended our grammar to distinguish between plural and singular sentences.
It required adding and changing many rules, especially noun and adjective phrases, determiner rules, and sentences in the present form.

The rules below are used to distinguish between plural and singular noun phrases

NNS → NP Cconj NP # we replaced the rule 'NP -> NP Cconj NP' in this rule for solving the problem in part2 that sentences (b) and (h)/(i) can create ungrammatical sentences.
NP → Det Noun
NP → Det Noun PP
NP → Det VowNoun
NP → Det VowNoun PP

NNS → PluralDet PluralNoun # for generating the phrases such as "the girls ate"
NNS → PluralDet PluralNoun PP # for generating sentences such as "the girls ate on the desk ."
NNS → PluralNoun # for generating the phrases such as "girls ate"
NNS→ PluralNoun PP # for generating sentences such as "girls ate on the desk ."

Some examples of NNS from our extended vocabulary are:

PluralNoun → presidents | girls | boys| women | men | children | people

Some examples of PluralDet from our extended vocabulary are:

PluralDet → the | all the | all | some | these | those | many

The rules below are used to generate past and present tense sentences of both the singular and plural form.

S_VBD → NP VP_VBD # for generating sentences in the single past form.
S_PRESENT→ NP VP_PRESENT_S # for generating sentences in the single present form.

S_VBD → NNS VP_VBD # for generating sentences in the plural past form.
S_PRESENT → NNS VP_PRESENT_P # for generating sentences in the plural present form.

The rules below are used to generate present tense sentences of singular and plural forms.

VP_PRESENT_S → Vbz ADJP # for generating sentences such as PART2(h) "Sally is lazy ."
VP_PRESENT_S → Vbz VowAdj # for generating sentences such as "Sally is old ."
VP_PRESENT_S → Vbz Vbg NP # for generating sentences such as PART2(i) "is eating a sandwich ."
VP_PRESENT_S → Vbz Vbg NNS # for generating sentences such as "is eating sandwiches ."
VP_PRESENT_S → Vbz Vbg_prep PP # for generating sentences such as "is eating on a desk ."
VP_PRESENT_S → Vbz NP # for generating sentences such as PART2(j) "... sally is a sandwich ."

VP_PRESENT_P → Vbp PADJP # for generating sentences such as "women are smart ."
VP_PRESENT_P → Vbp Vbg NP # for generating sentences such as "the girls are eating a sandwich ."
VP_PRESENT_P → Vbp Vbg NNS# for generating sentences such as "the girls are eating sandwiches ."
VP_PRESENT_P → Vbp Vbg_prep PP # for generating sentences such as "the girls are eating on a desk ."
VP_PRESENT_P → Vbp NNS # for generating sentences such as "are sandwiches ."

The rules below are used for supporting plural and singular that-clause

SBAR → Sconj NP Vbz NP # for generating sentences such as PART2(j) "that Sally is a sandwich ."
SBAR → Sconj NNS Vbp NNS # for generating sentences such as "that people are sandwiches ."
S_VBD → Prp Vbd_np_sconj NP SBAR # for generating sentences such as PART2(e) "it perplexed the president that a sandwich ate Sally ."
S_VBD → PluralPrp Vbd_np_sconj NP SBAR # for generating sentences such as "they perplexed the president that a sandwich ate Sally ."
S_VBD → PluralPrp Vbd_np_sconj NNS SBAR # for generating sentences such as "they perplexed men that a sandwich ate Sally ."
PluralPrp → they

The rules below are used for supporting plural and singular adjective phrases - In English, the adjectives are not changed to a plural form, but the determiner rules may change.

NP → Det ADJP Noun # for generating sentences such as PART2(f) "the very very very perplexed president ate a sandwich ."
NP → Det ADJP Noun PP # for generating sentences such as "the very delicious pickle under the desk ate a sandwich ."

NNS → PluralDet PADJP PluralNoun # for generating sentences such as "the very very very perplexed boys ate a sandwich ."
NNS → PluralDet PADJP PluralNoun PP # for generating sentences such as "some very delicious people under the desk ate a sandwich ."
NNS → PADJP PluralNoun # for generating sentences such as "very very very perplexed boys ate a sandwich ."
NNS → PADJP PluralNoun PP # for generating sentences such as "very delicious people under the desk ate a sandwich ."

ADJP → ADVP Adj | ADVP VowAdj | Adj
PADJP → ADVP PluralAdj | PluralAdj # PluralAdj = VowAdj **UNION** Adj

In addition, we added rules for supporting questions in part 4 to have singular and plural forms.

In the following phenomena, plural and singular forms are supported as well.

Note: We have mentioned the main rules - additional rules may be in the grammar5 file. (We extended the vocabulary as well).

Sentences in the form: has\have + NP\NNS(plural-noun)

The following rules are used for supporting sentences in the common form of NP\NNS + has\have + NP\NNS

VP_PRESENT_S → has NP # we added this rule for generating sentences such as "Sally has a sandwich ."
VP_PRESENT_S → has NNS # we added this rule for generating sentences such as "Sally has sandwiches ."

VP_PRESENT_P → have NP # we added this rule for generating sentences such as "people have a sandwich ."
VP_PRESENT_P → have NNS # we added this rule for generating sentences such as "people have sandwiches ."

Sentences in the form: ‘has\have to’ + base-form verb + NP\NNS

The following rules are used for supporting sentences in the common form of NP\NNS + has\have + to + base verb + NP\NNS

VP_PRESENT_S → has to Vp_intran # we added this rule for generating sentences such as "Sally has to eat ."
VP_PRESENT_S → has to Vp_prep PP # we added this rule for generating sentences such as "Sally has to stay with the president ."
VP_PRESENT_S → has to Vp_np_sconj NP SBAR # for generating sentences such as "Sally has to perplex the president that a sandwich ate liat ."
VP_PRESENT_S → has to Vp_np_sconj NNS SBAR # for generating sentences such as "Sally has to perplex presidents that a sandwich ate liat ."

VP_PRESENT_P → have to Vp_intran # we added this rule for generating sentences such as "people have to eat ."
VP_PRESENT_P → have to Vp_prep PP # we added this rule for generating sentences such as "Sally and liat have to stay with the president ."
VP_PRESENT_P → have to Vp_np_sconj NP SBAR # for generating sentences such as "Sally and liat have to perplex the president that a sandwich ate liat ."
VP_PRESENT_P → have to Vp_np_sconj NNS SBAR # for generating sentences such as "Sally and liat have to perplex presidents that a sandwich ate liat ."

Sentences in the present perfect form
Sentences in the form: should/could + have + V3(Vbn)

VP_PRESENT_S → has VP_VBN # present perfect form
VP_PRESENT_P → have VP_VBN # present perfect form

VP_VBD → MdPast have VP_VBN # should/could + have + (past participle)

VP_VBN → Vbn NP # for generating sentences such as "a girl has written a letter .", "a girl should have written a letter ."
VP_VBN → Vbn_prep PP # for generating sentences such as "liat has worked on every proposal on the desk ." , "liat should have worked on every proposal on the desk ."
VP_VBN → Vbn NNS # for generating sentences such as "a girl has writen letters .", "a girl should have writen the letters ."
VP_VBN → Vbn_intran # we added this rule for generating sentences such as "madona has eaten .", "madona could have eaten ."
Vbn → Vbn Cconj Vbn # we added this rule for generating sentences such as "madona has wanted and kissed the boy .", "madona could have wanted and kissed the boy ."

Some examples of Vbn from our extended vocabulary are:
Vbn → pickled | perplexed | understood | gotten | written | driven