# NLP course Assignment 3: Grammar Engineering

Daniel Bazar 314708181

Lior Krengel 315850594

## Part 1 | Weights

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 so many long sentences mainly because of some recursive grammar rules.   
   Specifically, the rule: NP → NP PP is responsible for that phenomena. Since every rule in this part has the same weight, and we start from the rule S → NP VP, then, we have 50% chance of choosing the above rule (because there are two rules for NP). In addition, the VP rule is also derived to a rule that includes NP. Moreover, the nonterminal PP that is derived from the NP rule contains in itself the nonterminal NP which means that NP is called at least twice.
2. **The grammar allows multiple adjectives, as in: “the fine perplexed pickle”. Why do the generated sentences show this so rarely? discuss.**

Generated sentences show multiple adjectives so rarely because in order to get such an event we need to derive the rule Noun → Adj Noun twice. This rule is one of 6 rules with Noun in the LHS, and all rules have the same weight, so the probability for it is just 1/6.

1. **The grammar format allows specifying different weights for different rules. Which numbers must you modify to fix the problems in (1) and (2), making the sentences shorter and the adjectives more frequent? Verify your answer by generating from the grammar. Discuss your solution (which rules did you modify, and why).**

First, to make sentences shorter we need to handle the problem with the NP → NP PP rule we described in (1). The problem was that this rule is chosen too often. so we increased the weight of the second NP rule (NP → Det Noun) to 5 making the relative ratio between the two rules in favor of the “non-recursive” one.

Second, to make multiple adjectives more frequent, we took the same approach, and we increased the weight of the Noun → Adj Noun rule to 5 to make it more frequent.

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

In order to make sentences more natural, we have adjusted the weights of the rules. This adjustment reflects the likelihood of word distribution. To estimate this, we have adjusted the weights of verbs and adjectives based on the nouns they are commonly paired with. For each verb or adjective, we have assigned a weight based on how many nouns can appear with it in the same context. For instance, for the verb 'ate', we assigned the weight 2 because it can connect with two nouns out of five (one can 'ate' a sandwich and a pickle, but not a president, a chief of staff, or a floor). Similarly, we determined the weights of the nouns based on the verbs and adjectives they are commonly paired with. We didn't change the weights of the determiners and prepositions because they are likely uniformly distributed.   
We have also adjusted the weights of the rules used to create full sentences. We increased the weight of a sentence ending with a period (".") because it is more likely than sentences ending with an exclamation mark ("!") or a question mark ("?").

## Part 2 | Extending the Grammar

We made modifications to the grammar in order to generate the types of phenomena illustrated in the given sentences. All explanations can also be seen in the “grammar2” file.

In order to solve this part problem, first we tagged each word in each sentence with the respective part-of-speech. Then we built a syntax tree and from it, we derived the below rules.

For each sentence, we provide the corresponding modifications:

* 1. NP → Nnp – to support proper nouns (people, locations, organization, etc.)
  2. Nnp → Sally – adding “Sally” to vocabulary as Nnp
  3. NP → NP Cc NP – to support coordinating conjunction (“…and…”)
  4. VP → Verb Cc VP – to support conjunction between verbs (“…and…”)
  5. Cc → and – adding “and” to vocabulary as Cc

To respect the *subcategorization frame of verbs* we reclassified the ‘Verb’ preterminal as transitive verbs (V1) and we added another preterminal called ‘V0’ as intransitive verbs.

* 1. VP → V0 – to support intransitive verb phrase
  2. V0 → sighed – adding “sighed” to vocabulary as V0
  3. VP → Verb Sc S – to support subordinating conjunction (“<dependent clause> *that* <independent clause>”)
  4. Verb → thought – adding “thought” to vocabulary as Verb
  5. Sc → that – adding “that” to vocabulary as Sc
  6. NP → Prp – to support personal pronoun (substitute to person like I, you, and *it*)
  7. NP → NP Sc S – to support subordinating conjunction with NP
  8. Verb → perplexed – adding “perplexed” to vocabulary as Verb. It’s Important to note that “perplexed” has already been used but in different meaning as Adj.
  9. Prp → it – adding “it” to vocabulary as Prp (personal pronoun as described above)

1. + (h)

After reaching sentence (h) we realized a strong connection between the two sentences, so we made some adjustments. Therefore, the final rules are derived from the sentences together.

* 1. NP → Det ADJP Noun – to support adjective phrase.
  2. ADVP → Rb – to support adverb phrase
  3. ADVP → Rb ADVP – to support recursive adverb phrase (“very very…”)
  4. Rb → very – adding “very” to vocabulary as Rb (adverb)
  5. VP → Vbz ADJP – to support 3rd person singular present (“is…”)
  6. ADJP → Adj – to support adjective phrase
  7. ADJP → ADVP Adj – to support modifying adjectives
  8. Adj → lazy – adding “lazy” to vocabulary as Adj
  9. Vbz → is – adding “is” to vocabulary as Vbz (new preterminal, Verb, 3rd person singular present)
  10. VP → Verb PP – to support PP (providing additional information) also to Verb
  11. Verb → worked – adding “worked” to vocabulary as Verb
  12. Noun → proposal – adding “proposal” to vocabulary as Noun
  13. Noun → desk – adding “desk” to vocabulary as Noun

Described above at (f)

* 1. VP → Vbz Vbg NP – to support a form of present progressive
  2. Vbg → eating – adding “eating” to vocabulary as Vbg (Verb, gerund or present participle. Ending with *-ing*)

Important note: we are asked in this part to generate *all* sentences given. In this, the word “sally” is written with a lowercase “s”. Despite this, we referred to it like the Nnp word “Sally” and we didn’t make another terminal “sally”.

* 1. VP → Vbz NP – to support present simple

**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.**

In sentence (b) we created rules for conjunction between phrases and in sentences (h)/(i) we created rules for linking (“is”). The potential grammatical issue arises when attempting to combine sentence structures (b) and (h)/(i). example for ungrammatical sentence: "Sally and the president is lazy."   
The problem is when coordinating "Sally and the president" it’s a plural subject. However, combining the linking “is”, leading to a grammatical disagreement in number. The correct form would be "Sally and the president are lazy".

## Part 3 | Tree Structures

We added another optional command line switch -t that generates the sentence tree structure.

All modifications can be seen in the code *generate.py.* specifically, we mainly modified the function *gen*.