NLP HW1 Written Answers – Taran Agarwal, Matthew Blackburn, Jehan Boer

1.4

1. **Provide 10 random sentences generated from your script.**
   - 1.1. the chief of staff on the president kissed a pickle with every president under every perplexed president with every chief of staff under a chief of staff on a chief of staff under every chief of staff .
   - 1.2. is it true that every floor understood every sandwich ?
   - 1.3. is it true that every delicious floor wanted every fine fine sandwich on the pickle on every sandwich ?
   - 1.4. a chief of staff kissed a floor on every sandwich on a chief of staff !
   - 1.5. every president pickled every floor on the sandwich in the chief of staff in every pickled president on the sandwich on every pickle in every pickle on a pickle in a president in a floor under every floor under a pickle on a chief of staff .
   - 1.6. s it true that a delicious floor kissed a president under the pickle ?
   - 1.7. the floor ate the sandwich on every delicious sandwich .
   - 1.8. a chief of staff kissed the sandwich on a fine fine floor on every sandwich .
   - 1.9. is it true that a chief of staff pickled every perplexed perplexed president ?
   - 1.10. the chief of staff wanted a fine president in every pickled floor .

2. **Provide 2 random sentences generated from your script, using --tree to show their derivations.**

   2.1.
```
(ROOT is
it
        true
        that
        (S (NP (Det the)
              (Noun pickle))
           (VP (Verb wanted)
              (NP (Det every)
                 (Noun (Adj pickled)
                    (Noun sandwich)))))
        ?)
```

   2.2.
```
(ROOT (S (NP (NP (NP (Det the)
                     (Noun pickle))
                 (PP (Prep on)
                    (NP (NP (Det every)
                            (Noun pickle))
                        (PP (Prep under)
                           (NP (Det a)
                              (Noun sandwich)))))))
          (PP (Prep under)
```

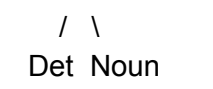
(NP (Det every)
(Noun chief
of
staff))))
(VP (Verb wanted)
(NP (Det every)
(Noun sandwich))))
.)

3. **As in the previous question, but with a --max expansions of 5.**
   3.1. (ROOT is
   it
   true
   that
   (S (NP (Det every)
   (Noun floor))
   ...)
   ?)
   3.2. (ROOT is
   it
   true
   that
   (S (NP (NP (Det every)
   ...)
   ...)
   ...)
   ?)

2

2.1

1. Sentences are expanded by nonterminal and preterminal expansions, so when we reach a value that will expand, we can think of the value as the head of a tree. Example:



Because the sample function runs recursively, a nonterminal expansion creates another layer of depth. Long sentences are a result of getting lots of nonterminals/preterminals. We can specify the rule on line 38, NP -> NP PP, as the rule responsible for long sentences. This expansion exponentially increases the number of expansions and guarantees two nonterminal branches, where one branch, NP, has a 0.5 chance of creating another nonterminal, and the other branch, PP, guarantees a nonterminal.

2. Based on grammar.gr's rules, when we reach a noun, there are 5 terminal options and 1 option to expand noun to **adj noun**. The chance of getting an adjective in front of a noun is 1/6 (since all rules have equal probability in this grammar).

$$P(\text{Adj Noun} \mid \text{Noun}) = \frac{1}{6}$$

To get n adjectives preceding a noun would have the low probability of (1/6)^n, hence making it a rare occasion once we expand to a noun.

3. Lines 38 and 40 must be modified.

Line 38: NP -> NP PP
Line 40: Noun -> Adj Noun

Decreasing the probability of line 38 will make sentences shorter, while increasing the probability if 40 will result in more adjectives preceding nouns.

4. We can adjust the probabilities of nouns, adjectives, verbs, and S types so they match frequency based on the English language. This means decreasing the probabilities of verbs like pickled and adjectives like pickled & perplexed.

5.
   5.1. every delicious delicious perplexed president understood the president .
   5.2. a fine chief of staff ate a delicious president !
   5.3. the delicious delicious president pickled a president under every perplexed floor !
   5.4. every president pickled a fine delicious president with every delicious delicious sandwich .
   5.5. every delicious floor kissed a floor with every perplexed floor !
   5.6. is it true that a floor understood the president ?
   5.7. every floor wanted a fine delicious chief of staff !
   5.8. is it true that a delicious chief of staff wanted a sandwich ?
   5.9. the sandwich ate a president !
   5.10. a fine pickle understood a floor .

2.3

1. We made quite a few modifications in order to have a grammar that could construct the examples in 2.2. We can run by them one by one:
   a. **Verbs:**
      i. **Transitive/Intransitive**: We introduced the idea of transitive (needs an object) and intransitive (stands alone with no object) verbs to capture the behavior of certain verbs. For example, sighed an intransitive verb and perplexed is a transitive verb. Most verbs in the grammar were both

transitive and intransitive (can exist alone or with a subject), so we denoted most verbs as just verbs, while the ones that were strictly transitive or intransitive as Strict. This allowed us to form sentences like question 3 as well as avoid the erroneous sentence mentioned in section 2.2

   ii. **Causative Verbs:** We introduced the idea of causative verbs, verbs that cause a change of state / cause an action. This allowed us to form sentences like example 5.
   iii. **Reporting Verb:** We introduced reporting verbs. A reporting verb is a verb used to introduce thoughts. This allowed us to construct sentences like example 4.

b. **Conjunctions:**
   i. **Regular conjunctions:** We introduced the idea of conjunctions so that we could connect phrases, whether they were noun phrases or verb phrases. This helped us construct sentences like example 2.
   ii. **Subordinating conjunctions:** Subordinating conjunctions connect a dependent clause to an independent clause. This helped us construct sentences like example 5.

c. **Adverbs:**
   i. We added adverbs like ‘very’ to help describe adjectives. This allowed sentences like example 7 to be constructed.

d. **Embedded clause:**
   i. We introduced the idea of embedded clauses (dependent/subordinate clauses) to allow the construction of dependent clauses which could be used to construct sentences like 4,5,6.

e. **Nouns**
   i. **Locative Nouns:** We introduced the idea of Locative Nouns. Locative Nouns are simply nouns that denote a location. This was necessary for prepositional phrases, where we would describe where something was relative to a place. This allowed us to construct sentences like example 8.
   ii. **Proper Nouns:** To handle names (like Sally) we introduced Proper Nouns. Proper Nouns act similar to regular nouns, they just don’t need determiners. This allowed us to generate sentences with names as subjects (like examples 1, 2 & 6).
   iii. **Dummy subjects:** We introduced the idea of a dummy subject which allows there to be a grammatical subject when no real subject is needed, like in example 5.

f. **Phrases/Sentences:**
   i. Using these new ideas, we added new expansions for Verb, Noun, and Prepositional Phrases, as well as sentences to construct a new variety of sentences, including all of the examples from 2.2.

2. 10 random sentences generated with grammar3:
   a. the president perplexed a pickle !
   b. Matthew Blackburn with the desk pickled Sally .

c. that Taran Agarwal worked Jehan Boer on a desk perplexed Matthew Blackburn !
d. the chief of staff thought and thought !
e. Taran Agarwal perplexed every president .
f. Taran Agarwal under a floor on Matthew Blackburn on every floor in the floor perplexed Jehan Boer .
g. Matthew Blackburn understood that every chief of staff kissed and perplexed Sally .
h. a president perplexed the sandwich !
i. Matthew Blackburn kissed .
j. a proposal ate !

Part 3

1.

a)

```
(ROOT (S  (NP (NP (Det every)
                  (Noun sandwich))
              (PP (Prep with)
                  (NP (NP (Det a)
                          (Noun pickle))))
                      (PP (Prep on)
                          (NP (Det the)
                              (Noun floor)))))
          (VP (Verb wanted)
              (NP (Det a)
                  (Noun president))))
.)
```

b)

In the original derivation, the grouping of the noun phrases made it refer to every sandwich being on the floor. In the second derivation, the pickle is referred to as on the floor rather than the sandwich. We care about the derivation of the sentence because different derivations of the sentence can mean different things.

2.

The parser does not always recover the original derivation that was intended by randsent. This happens especially in the different rules that allow for nesting such as with VPs and NPs, where the parser could create different groupings from the ambiguity that nesting causes.

A few examples have been generated by grammar3:

The sentence “the proposal ate a chief of staff and kissed a delicious very delicious chief of staff and perplexed every chief of staff and sighed with a proposal !” was generated in tree form like:

```
(ROOT (S (NP (Det the)
          (Noun proposal))
       (VP (VP (VP (Verb ate)
                (NP (Det a)
                   (Noun chief
                      of
                      staff)))
             (Conj and)
             (VP (VP (VP (Verb kissed)
                      (NP (Det a)
                         (Noun (Adj delicious)
                            (Noun (Adj (AdVerb very)
                                     (Adj delicious))
                               (Noun chief
                                  of
                                  staff)))))
                   (Conj and)
                   (VP (StrictTransVerb perplexed)
                      (NP (Det every)
                         (Noun chief
                            of
                            staff))))
                (Conj and)
                (VP (StrictInTransVerb sighed))))
          (PP (Prep with)
             (NP (Det a)
                (Noun proposal)))))
    !)
```

However, the parser gave the tree:

```
(ROOT (S (NP (Det the)
          (Noun proposal))
       (VP (VP (VP (Verb ate)
                (NP (Det a)
                   (Noun chief
                      of
                      staff)))
             (Conj and)
             (VP (Verb kissed)
                (NP (Det a)
                   (Noun (Adj delicious)
                      (Noun (Adj (AdVerb very)
```

```
                                (Adj delicious))
                            (Noun chief
                                 of
                                 staff))))))
        (Conj and)
        (VP (VP (StrictTransVerb perplexed)
                (NP (Det every)
                    (Noun chief
                         of
                         staff)))
            (Conj and)
            (VP (VP (StrictInTransVerb sighed))
                (PP (Prep with)
                    (NP (Det a)
                        (Noun proposal)))))))
    !)
```

The original tree is made up of recursive nesting of VPs connected by the conjunction “and” and includes the verbs “ate”, “kissed”, “perplexed”, and “sighed”. The parser correctly captures the phrase until after kissed, where in the original tree, “perplexed” and “sighed” are nested inside a recursive VP with conjunctions, while in the parser’s result, there is a flatter structure that groups them under their own VP.

In another output by grammar3, the sentence “a proposal perplexed every very very very very very very very very very very very very very very fine chief of staff .” was generated in tree form like:

```
(ROOT (S (NP (Det a)
             (Noun proposal))
         (VP (StrictTransVerb perplexed)
             (NP (Det every)
                 (Noun (Adj (AdVerb (AdVerb (AdVerb very)
                                            (AdVerb very))
                                    (AdVerb (AdVerb very)
                                            (AdVerb very)))
                            (Adj (AdVerb very)
                                 (Adj (AdVerb (AdVerb (AdVerb very)
                                                      (AdVerb very))
                                              (AdVerb (AdVerb very)
                                                      (AdVerb (AdVerb very)
                                                              (AdVerb (AdVerb very)
                                                                      (AdVerb very)))))
                                      (Adj (AdVerb (AdVerb (AdVerb very)
                                                           (AdVerb very))
```

```
                                    (AdVerb very))
                                (Adj fine)))))
                    (Noun chief
                        of
                        staff)))))
    .)
```

However, the parser gave the tree:

```
(ROOT (S (NP (Det a)
             (Noun proposal))
         (VP (StrictTransVerb perplexed)
             (NP (Det every)
                 (Noun (Adj (AdVerb (AdVerb (AdVerb very)
                                            (AdVerb (AdVerb very)
                                                    (AdVerb (AdVerb very)
                                                            (AdVerb (AdVerb (AdVerb (AdVerb very)
                                                                                    (AdVerb very))
                                                                            (AdVerb (AdVerb very)
                                                                                    (AdVerb (AdVerb very)
                                                                                            (AdVerb very))))
                                                                    (AdVerb (AdVerb (AdVerb very)
                                                                                    (AdVerb very))
                                                                            (AdVerb (AdVerb very)
                                                                                    (AdVerb (AdVerb very)
                                                                                            (AdVerb very))))))))
                                                                                            (AdVerb very))
                                                                                        (Adj fine))
                                                                                    (Noun chief
                                                                                        of
                                                                                        staff)))))
    .)
```

In this case, the repetition of “very” comes from the rule Adverb -> Adverb Adverb, which allows for a lot of nesting. The difference in the result of the parser is that it flattened the structure by grouping some of the nesting and separating them.

3.
This noun phrase simply alternates noun phrases and prepositions (the sentence can be broken down as NP Preposition NP Preposition NP Preposition NP). In the formation of this sentence, besides the first noun phrase, “every sandwich”, all noun phrases are generated through prior ones through the rule NP -> NP PP. The different formations of the noun phrase come from the previous noun phrases that following ones are generated from. The different formations are as follows:

1. “a pickle”, “the floor”, and “the chief of staff” are all generated from “every sandwich”
2. “a pickle” is generated from “every sandwich”, “the floor” is generated from “a pickle”, and “the chief of staff” is generated from “the floor”
3. “a pickle” is generated from “every sandwich” and “the floor” and “the chief of staff” are generated from “a pickle”
4. “a pickle” and “the floor” are generated from “every sandwich” and “the chief of staff” is generated from “a pickle”
5. “a pickle” and “the floor” are generated from “every sandwich” and “the chief of staff” is generated from “the floor”

Therefore, there are 5 ways to analyze the noun phrase. If we run: ./parse -g grammar.gr -s NP -c, with -s NP making us start at an NP and -c showing the number of parses, we get 5 which confirms our answer.

4.

In grammar.gr, when running this, we found that the longer winded sentences with lots of recursion tend to have more pareses. As we discussed before, the more recursion allows for ambiguity with groupings, leading to more parses. For example, this highly recursive sentence, “is it true that a perplexed president kissed every chief of staff on a sandwich on every pickle in the pickle under every perplexed chief of staff under every president in the president with every fine chief of staff ?” has 429 parses. On the other hand, a more simply formed sentence “a president wanted every delicious chief of staff” only has 1 parse. In grammar3.gr, we observe similar behavior, although grammar 3 has more rules that allow for recursion.

5.

a)

- This probability, p(best_parse) comes from multiplying together all of the probabilities at each branch in the tree of best_parse.
- p(sentence) is equivalent to p(best_parse) because there is only one derivation of the sentence
- p(best_parse | sentence) = 1 also because there is only 1 derivation of the sentence

b)

p(best_parse | sentence)= 0.5 because, as we showed earlier in 3.1 a), there are 2 derivation trees of this sentence, both equally probable, so the probability of a given parse being best_parse is ½ or 0.5.

c)

The cross entropy can be estimated through a formula that adds the base 2 log of the p(best_parses) all divided by the total number of words including punctuation, all negated. In this case, it would be:

$cross\ entropy\ =\ ($

$-\ (log_2 5.144\ *\ 10^{-5})\ -\ (log_2 6.202\ *\ 10^{-10}))/18 \approx\ -\ (-\ 43.833/18)\ =\ 2.435$

d)

Perplexity per word is calculated as $2^{cross\ entropy}\ =\ 2^{2.435} \approx 5.408$

e)

“the president ate .” is not a valid sentence in the grammar since a VP has to be followed by a NP, and the sentence ends in a VP with no NP following. Therefore p(best_parse) would be 0, and log0 is infinity, so cross entropy would be undefined.

6.
a)
We used the command – python3 randsent.py -g grammar2.gr -n 10000 | ./parse -P -g grammar2.gr – to estimate the entropy of grammar2. We ran it 10 times (values in bits: 2.120, 2.120, 2.121, 2.119, 2.119, 2.119, 2.121, 2.120, 2.118, 2.120) for an average entropy of 2.1197.

b)
We ran the same command on grammar 3 (values in bits: 2.136, 2.138, 2.137, 2.134, 2.138, 2.134, 2.135, 2.133, 2.139, 2.134) for an average entropy of 2.1358. It makes sense that the entropy is higher for grammar 3 as there are more rules and therefore more opportunity for more complex sentences, so the probabilities of best parse would be lower which would lead to entropy being higher.

c)
In grammar.gr, the sentences often hit max expansions, denoted as … which cannot be parsed by the parser. This comes from higher probability recursive rules like NP -> PP which leads to deep recursions. Because of this, since the parser can’t parse it, the p(best_barse) comes out as NaN which leads the cross entropy to be infinity.

7)
Evaluating grammar2 with a grammar parser gives us a cross entropy of 2.423 bits, whereas evaluating grammar2 with a grammar3 parser gives us a cross entropy of 3.057. It makes sense that both entropies are higher than the 2.1197 found in 3.6a. It also makes sense that grammar3 parser is much higher than grammar’s since grammar2 and grammar are very similar, differing only in probabilities, whereas grammar3 has much more rules, words, and complexity which would create higher cross entropy.

4.

1. Phenomena Chosen and Changes:
    a. The first phenomena chosen was e) singular vs plural agreements. Our changes were as such:
        i. **Nouns:**
            1. **Singular vs plural nouns:** In order to create singular and plural agreements, we needed to indicate which nouns were plural and which were singular. We created a new set of nouns, denoted as either singular or plural: SingularNoun, PluralNoun.
        ii. **Verb:**
            1. **Singular vs plural verbs:** In order to create these singular and plural agreements, we also needed to denote which verbs were singular and which verbs were plural. Mixing a plural noun with a

singular verb or vice versa would give incorrect, unnatural language: ex. The citizens chooses. VerbSingular and VerbPlural helped map these verbs and nouns to each other to create these agreements.

iii. **Determiners:**
    1. **Singular vs plural determiners:** In addition to nouns and verbs, determiners are also specified based on whether they are plural or singular. We denoted determiners as either singular or plural to help map these plural and singular agreements. ‘The citizens’ is good while ‘a citizens’ is bad.

iv. **Verb Phrases, Noun Phrases, Sentences:**
    1. The last step in mapping the plural and singular agreements to each other was by denoting all the different phrase types as plural or singular. A plural noun phrase combined with a singular verb phrase would not make sense, and so it was important to ensure that plural noun phrases would combine with only plural verb phrases. New sentence types were added to match these singular/plural agreements. Additional steps were taken within Verb and Noun phrases to ensure all the plural and singular edge cases were handled to construct a variety of different plural / singular agreeing sentences.

b. The second phenomena we chose was g) to use appositives. Our changes looked as follows:

i. **Relative clauses:**
    1. One of the types of appositives in the examples were relative clauses, dependent clauses that give more information about a main clause. These clauses helped us generate appositives such as ‘who ate a sandwich’

ii. **Numerical phrases:**
    1. Pretty simple, but we made numerical phrases to act as a type of appositives. We added variety with the numbers that could be chosen.

iii. **DetAppos:**
    1. For appositives like ‘the chief of staff’, we realized that only certain determiners worked, and so we created a new category of determiners specifically for appositives.

iv. **Appositives:**
    1. We added appositives to handle two cases: the case where an appositive was in the middle of a sentence and the case where an appositive was at the end of a sentence. Handling these two cases was very different:
        a. **End of sentence case:**
            i. In order to handle the EOS appositive case (where we don’t want the trailing comma, we had to make

our way up the tree all the way to the root. Essentially, for a sentence to end with an appositive (like the first example for g), we simply stuck an appositive at the end of a real sentence. The key distinction here was that we only wanted one comma, so **, Appos** instead of , **Appos ,.**

b. **Middle of sentence case:**
    i. The middle of sentence case was also relatively hard. One middle of sentence appositive would not have been that difficult, but we needed to find a way to chain them to match the second example for g. We moved up the tree all the way to noun phrases and introduced something called a complex noun phrase (**ComplexNP)**. A subject, followed by **n** appositives still acts as a noun, and so we catered to this idea. A complex noun is simply a noun phrase that is described by an **ApposChain** (chain of appositives). The key here was to avoid repeated commas (Sally, the chief of staff,, the president) that would occur if you just put two appositives next to each other. The ApposChain recursively and gracefully handled this chain and allowed there to be anywhere from 1 to n appositives describing a noun (complex noun). We created a new sentence type to describe sentences with ComplexNouns, simply **ComplexNP VB.**
c. **Generally:**
    i. Generally, all appositives were either relative clauses, numerical phrases, or noun phrases. To avoid the unnaturalness of potentially long complex noun phrases, we limited the appositive noun phrases to be: **DetAppos Noun.** Appositives don't sound great when they are too long or when they are Proper Nouns.

5. Extra Credit

1. To make our grammar more interesting, we added the notion of auxiliary verbs (helping verbs) so that we could generate sentences that comment on the **state** of a subject. We handled auxiliary verbs for all tenses as well (**past, present, future).** Interestingly, future states are handled different from present and past states in the English language, so we had to work accordingly with this. All aux verbs were implemented through verb phrases.

a. **Present and past states:**
    i. To handle present and past states, we added auxiliary verbs for each of these states, denoted by their tense.
    ii. Additionally, present and past states are objects, so noun phrases. Therefore we created the expansion for VP to become AuxP (present/past auxiliary verbs) NP. He is the chief of staff, he was the president etc.
b. **Future states:**
    i. Future states are **very** interesting as they do not have any similar behavior to present and past states. *He is the president, he was the president, he will become the president*. Will become acts as the verb which very different compared to is and was. To make things interesting, we made future states defined by Verb Phrases as opposed to noun phrases. *He will eat a sandwich.* Therefore, we added the AuxVerbFutureState which is always followed by a verb phrase (we created the expansion VP -> AuxF VP.
c. **Potential Next Steps:**
    i. The ‘will became’ future tense of is and was introduces a lot of complexity to the grammar. It would be interesting as a next step how we could incorporate ‘became’ into the grammar so that it allows for future states with Noun Phrases as opposed to verb phrases.