Justin Bravo and Griffin Montalvo

NLP Homework 1 Questions

# Questions 1.4:

1. Provide 10 random sentences generated from your script.
    - the sandwich wanted a chief of staff under a sandwich with every chief of staff .
    - is it true that the perplexed sandwich on the president in a chief of staff in every sandwich in every sandwich with every pickle on a sandwich in every fine perplexed pickle with the chief of staff under the president with the sandwich on every floor under every floor on a sandwich on the pickle with the pickle in every fine pickle with the fine floor on every chief of staff under every pickle in the sandwich with a floor in the pickle on the sandwich with every pickle under a pickle in the pickle with a chief of staff in every sandwich with every pickle under a pickle with every chief of staff under a floor on the pickle under the president with every president with every pickled chief of staff in every delicious chief of staff in the president with every president under every pickled sandwich with the delicious floor in every sandwich under the delicious pickle in every president on every chief of staff wanted every delicious fine president with every floor with a chief of staff ?
    - the pickle on every perplexed pickled sandwich with every floor on the fine floor on the pickle pickled the floor !
    - a pickle with a sandwich with the chief of staff understood every chief of staff on every president on a president with the pickle in every sandwich with every chief of staff under every president under every pickle under a delicious president under every sandwich under the president on a floor with every chief of staff in the floor with the delicious floor on a sandwich under every president under the sandwich with the pickle under the pickle in a president on a pickle in every perplexed sandwich on a chief of staff under a sandwich with the chief of staff with every chief of staff under a sandwich in a pickle with every floor with the pickle in the delicious chief of staff on the floor with a floor with every sandwich on the pickle under the perplexed floor on a pickle on the perplexed president under every pickle with a fine sandwich under the floor in a president with the pickle with every pickle under a delicious sandwich with every chief of staff in every sandwich under every floor under a delicious sandwich under a chief of staff with the pickle on every president in every pickled chief of staff in every chief of staff with every chief of staff in the president under every pickle on the floor in every pickle on the fine sandwich with every sandwich on a pickle with the pickle ... ...
    - a pickled sandwich ate every chief of staff !
    - the delicious president in every president on every president with the floor with the delicious chief of staff pickled the chief of staff with every floor in every pickle .
    - is it true that the sandwich with the pickled sandwich ate a pickle ?
    - the pickle wanted a president .
    - the president under the pickle under every delicious perplexed president wanted the pickle .

- is it true that the floor kissed a president ?

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

```
(ROOT (S (NP (NP (NP (NP (NP (Det the)
                             (Noun floor))
                         (PP (Prep in)
                             (NP (Det every)
                                 (Noun chief
                                       of
                                       staff))))
                     (PP (Prep on)
                         (NP (Det a)
                             (Noun (Adj fine)
                                   (Noun president)))))
                 (PP (Prep in)
                     (NP (Det the)
                         (Noun president))))
             (PP (Prep in)
                 (NP (Det a)
                     (Noun pickle))))
         (VP (Verb understood)
             (NP (NP (NP (Det the)
                         (Noun pickle))
                     (PP (Prep on)
                         (NP (Det the)
                             (Noun sandwich))))
                 (PP (Prep in)
                     (NP (NP (Det every)
                             (Noun floor))
                         (PP (Prep under)
                             (NP (Det the)
                                 (Noun sandwich))))))))
      .)
(ROOT (S (NP (Det a)
             (Noun president))
         (VP (Verb pickled)
             (NP (Det the)
                 (Noun (Adj delicious)
                       (Noun president)))))
      !)
```

3. As in the previous question, but with a --max expansions of 5.

```
(ROOT (S (NP (Det the)
              (Noun ...))
           ...)
        ...)
(ROOT (S (NP (Det the)
              (Noun ...
                   ...))
           ...)
        ...)
```

# Questions 2.1:

1. Why does your program generate so many long sentences? Specifically, what grammar rule (or
rules) is (or are) responsible and why? What is special about it/them?

Our program generates so many long sentences due to the combination of the recursive rules and also the structure of certain nonterminal expressions. Mostly the ones that involve noun phrase (NP) and prepositional phrase(PP) rules. A specific rule being: 1 NP   NP PP. This is a recursive rule because the nonterminal NP appears on both the left-hand side (LHS) and the right-hand side (RHS). Each time NP is expanded, it has the possibility of expanding into yet another NP, followed by a prepositional phrase, which can continue indefinitely until some limit is reached. Another rule being 1   PP Prep NP. The key issue here is that NP can recursively expand,
as we saw in the previous rule. Since PP calls NP this creates a cycle, where NP can recursively generate new NPs through the combination of NP → NP PP and PP → Prep NP. What's special about them is that the recursive nature of the rules means that there is no limit to how many times a noun phrase can actually expand so this creates really long sentences. Now the rule PP -> Prep Np will keep reintroducing NP into the sentence, which will just recursively call NP -> NP PP again and again, and this leads to nested prepositional phrases

2. The grammar allows multiple adjectives, as in the fine perplexed pickle. Why do your program’s sentences do this so rarely? (Give a simple mathematical argument.)

The reason why our sentences rarely contain multiple adjectives can be explained by the probability distribution of the grammar rules that currently exist. We specifically have to look at the rule that allows for multiple adjectives, which is the rule: Noun -> Adj Noun. Each time the program generates a new noun through the recursive rule, there are two possible outcomes: Recursion (applying Adj Noun again) or Termination(expanding to a terminal noun). In other words every time the rule is applied the recursive probability decreases because the number of possible expansions shrinks as the recursion depth increases. Therefore, the probability of having more than two adjectives in a row becomes small as the recursion continues. Furthermore each recursive expansion has a 50% chance of stopping after any given application, so the probability of generating a long sequence of multiple adjectives decreases significantly with each additional adjective.

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

We must change the numbers associated with two rules those being:
1 NP NP PP (This is the rule that causes long sentences) and 1 Noun Adj Noun (The rule that controls the presence of multiple adjectives is) We need to decrease the likelihood of the recursion that is associated with causing long sentences, so this would reduce the likelihood of the rule being chosen and therefore it will prevent overly long sentences. Now we need to increase the recursion that is associated with the presence of multiple adjectives, so this will make it more likely that another adjective is added before expanding to a terminal noun.

4. What other numeric adjustments can you make to grammar2.gr in order to favor more natural
sets of sentences? Experiment. Explain the changes.
We made the following changes to these rules:

* Changed all of the Weight of Terminal Noun Rules to 1.2 . We did this because it makes it more likely that sentences terminate earlier with a noun, reducing excessive recursion. Because adjectives are more frequent now these changes are ensuring that sentences don't end up with too many adjectives

* 0.8 PP Prep NP. We did this because although prepositional phrases are key to a sentence structure overusing them can make sentences very complex. Since we already reduced the recursion of noun phrases this change balances that by lowering the probability of prepositional phrases

* We changed the probability of some verbs to favor more common verbs like "ate" or "wanted"(so increased the probability of those verbs), then we lowered the probability of verbs like "pickled" and "understood" because we feel that those verbs make sentences little less natural

* We also changed the probability of some determiners. We increased "the" because it's a very common determiner in English. Then we decreased the probability of "every" because it's a lot less common than determiners such as "the" and "a"

5. Provide 10 random sentences generated with the grammar2.gr.

the president ate a perplexed perplexed pickle .
is it true that a floor pickled a pickle with every sandwich ?
is it true that the chief of staff ate every floor under the sandwich ?
a perplexed president ate the chief of staff in the floor !
a delicious president wanted the fine floor !
is it true that a chief of staff wanted every delicious sandwich ?
the president wanted the perplexed sandwich !
every sandwich ate every floor .
the perplexed sandwich wanted the president !
is it true that a chief of staff ate every sandwich ?

# Questions 2.3
9. Briefly discuss your modifications to the grammar.

We introduced pronouns like "it" and "Sally". The weights for determiners and noun phrases were fine-tuned to balance sentence complexity and avoid repetitive patterns like "Sally in Sally". We also introduced a distinction between transitive and intransitive verbs, which allowed for more natural sentence construction with complement causes and verb conjugations. We had to add more VP rules given the distinction
we made between TV and IV. We also added the new nouns and verbs that came with generating the sentences.

10. Provide 10 random sentences generated with grammar3.gr that illustrate your modifications.

the pickle ate Sally and kissed every proposal !
a sandwich sighed .
Sally sighed and worked .
the pickle and a chief of staff worked !
a chief of staff understood a very fine pickle !
a delicious floor ate Sally and pickled the delicious floor with the chief of staff .
every pickle and the president thought and sighed !
the very very very very pickled proposal wanted the floor and wanted every desk !
the chief of staff kissed the chief of staff !
Sally kissed the proposal !

# Questions 2.4:

- How is tree-structured, rule-governed behavior actually implemented in neural hardware?

It's explicitly stored in neural hardware. Neural networks and the brain learn patterns from exposure to data. These patterns are represented across many neurons. They allow neural networks to process language hierarchically and they exhibit rule-like behavior without actually needing grammar rules or trees

- Would artificial neural networks also be able to learn unnatural languages that don’t behave like this? Would human children?

It's a yes for artificial neural networks because they are pretty flexible in terms of learning patterns. As long as the data that is provided has consistent patterns, an artificial neural network could adapt and learn them. For human children it would be a much harder task and they would struggle with learning unnatural languages. This is due to their brains being predisposed to acquire natural languages that follow structure and hierarchy. Unnatural languages would not align with those mechanisms so it would make it very hard for human children to learn them.

- If so, then why do all natural languages still seem to use tree-structured sentences governed by rules? What guided their evolution so that they are the way that they are?

Natural languages are the way that they are due to the way our brains process information. At a very high level, without us even thinking about it, we form sentences in a tree-like manner, because there is a nearly infinite amount of sentences that we can form. These rules guide how we develop said sentences, and without them, sentences would just be a random assortment of rules with no context. A common set of communication rules needed to be developed in the first place in order to facilitate our evolution as human beings, and development arose because of our basic human desire for growth.

- The skeptic’s take: Are grammars even a good approach to modeling linguistic data? Or are they merely one kind of model that scientists (linguists) were predisposed to find?

While not the only model, we believe grammar is effective due to how thorough its set of rules can be. The rules leave little room for error, and the recursive nature allows for many sentences to be created.

– Is it a problem for the approach that a full English grammar—that is, a theory of English syntax—would be very large and hard to construct?

It certainly is a problem because it seems impractical to implement. the time and resources needed to build a full set of rules to outline a grammar would be strenuous.

– Is it a problem that people sometimes produce and understand ungrammatical sentences?

Yes it is a problem, because it is almost impossible to predict the unpredictable human mind. While grammar can cover grammatically correct sentences, it cannot cover the nuance (or laziness) of an grammatically incorrect sentence. With that being said, we think that it is possible to understand these grammatically incorrect sentences because of precedent grammar rules.

– What aspects of language do we miss when we focus on grammars?

Grammar defines the rules of how things should be said, but it does not focus on the meaning of what is being said. It can not depict differences between sentences that are the exact same, but have 2 completely different meanings. For example, "I saw her duck". This can either mean that "I was looking at her pet duck" (which itself has 2 meanings), or "I saw her physically move towards the ground". Grammar cannot depict the difference between these 2 meanings, it can only form a confusing sentence.

# Questions 3.1:
a) One derivation is as follows; what is the other?

In the original one we see both prepositional phrases "with a pickle" and "on the floor" attached to the noun sandwich. However the other derivation attaches "with a pickle" to the noun sandwich and "on the floor" to the noun "pickle".

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

b) There is a reason to care about which derivation was used because it affects the meaning of the sentence. In the first derivation noun is attached to both "with a pickle" and "on the floor" so this means that the sandwich itself is both accompanied by a pickle and located on the floor. However in the second one we are saying that a sandwich has a pickle, but the pickle is on the floor and not the sandwich.

# Questions 3.3:
2. Does the parser always recover the original derivation that was “intended” by randsent? Or does it ever “misunderstand” by finding an alternative derivation instead? Give a few examples and discuss.

Most of the time the parser will be able to fully recover the original derivation that was "intended" by the randset.
However, there are some cases where we can say the parser did not fully recover the exact derivation used by randsent.
An example is the following:
(Produced by randset)

```
(ROOT (S (NP (Det the)
             (Noun pickle))
         (VP (TV wanted)
             (NP (NP (NP (NP (Det the)
                             (Noun sandwich))
                         and
                         (NP (NP (PN Sally))
                             (PP (Prep with)
                                 (NP (NP (Det the)
                                         (Noun proposal))
                                     (PP (Prep on)
                                         (NP (Det the)
                                             (Noun chief
                                                   of
                                                   staff)))))))
                     and
                     (NP (Det every)
                         (Noun chief
                               of
                               staff)))
```

```
            (PP (Prep under)
               (NP (Det a)
                  (Noun president))))))
     !)
```

This was produced by the parser:
(ROOT (S (NP (Det the) (Noun pickle)) (VP (TV wanted) (NP (NP (Det the) (Noun sandwich)) and
(NP (NP (PN Sally)) (PP (Prep with) (NP (NP (Det the) (Noun proposal)) (PP (Prep on) (NP (NP (Det the) (Noun chief of staff)) and (NP (Det every) (Noun chief of staff)))) (PP (Prep under) (NP (Det a) (Noun president))))))))) !)

The difference: In the output from randsent, the “the sandwich” structure appears to be nested more deeply: (NP (NP (NP (NP (Det the) (Noun sandwich)), while in the parser output, the same structure is simpler: (NP (NP (Det the) (Noun sandwich)). The parser found a valid alternative that also fits the grammar but simplifies the nested structure. This can happen because grammars often allow multiple valid ways to generate the same surface-level sentence. The parser might optimize for simpler structures, while randsent tends to use more recursive rules, leading to deeper derivations. To further show that this is true here is another example:

This is generated by randset:

```
(ROOT (S (NP (NP (NP (Det the)
               (Noun (Adj perplexed)
                  (Noun floor)))
            and
            (NP (Det the)
               (Noun chief
                  of
                  staff)))
         and
         (NP (NP (NP (Det that)
                  (Noun president))
               and
               (NP (Det the)
                  (Noun desk)))
            (PP (Prep under)
               (NP (NP (Det the)
                     (Noun chief
                        of
                        staff))
                  and
                  (NP (Det the)
                    (Noun (Adj perplexed)
                          (Noun pickle)))))))
        (VP (TV wanted)
            (NP (Det the)
                (Noun sandwich))
            and
            (TV ate)
            (NP (Det the)
                (Noun (Adj perplexed)
                      (Noun desk)))))
    !)
```

Now here is the same sentence with the parser:
(ROOT (S (NP (NP (Det the) (Noun (Adj perplexed) (Noun floor))) and (NP (NP (Det the) (Noun chief of staff)) and (NP (NP (Det that) (Noun president)) and (NP (NP (Det the) (Noun desk)) (PP (Prep under) (NP (NP (Det the) (Noun chief of staff)) and (NP (Det the) (Noun (Adj perplexed) (Noun pickle))))))))) (VP (TV wanted) (NP (Det the) (Noun sandwich)) and (TV ate) (NP (Det the) (Noun (Adj perplexed) (Noun desk))))) !)

As you can also see in this example: ROOT (S (NP (NP (Det the) -> by parser and then (ROOT (S (NP (NP (NP (Det the) -> by randset and this shows up again in the same sentence (NP (NP (Det that) (Noun president)) -> by parser and then (NP (NP (NP (Det that)(Noun president)) -> by randsent. Once again this further shows that the the parser will optimize for simpler structures and then the randset tends to use more recursive rules and that just leads to more derivations.

How many ways are there to analyze the following noun phrase under the original grammar? (That is, how many ways are there to derive this string if you start from the NP symbol of grammar.gr?) every sandwich with a pickle on the floor under the chief of staff

We have three prepositional phrases in the sentence. Those being "with a pickle", "on the floor" and also "under the chief of staff". Each PP can attach to different noun phrases. So the number of ways to parse the phrase depends on how many combinations of attachments we can form.

1. All PPs attach to "every sandwich."
2. "with a pickle" attaches to "every sandwich" the others attach to "pickle."
3. "with a pickle" attaches to "sandwich," "on the floor" attaches to "sandwich," and "under the chief of staff" attaches to "pickle."
4. "with a pickle" and "on the floor" attach to "sandwich," "under the chief of staff" attaches to "floor."
5. "with a pickle" attaches to "sandwich," "on the floor" attaches to "pickle," and "under the chief of staff" attaches to "floor."

So in total there are 5 ways to parse the noun phrase “every sandwich with a pickle on the floor under the chief of staff” according to the grammar. And if we check this with the parser options we see that we are correct.

4. By mixing and matching the commands above, generate a bunch of sentences from grammar.gr, and find out how many different parses they have. Some sentences will have more parses than others. Do you notice any patterns?

grammar.gr: I've noticed some patterns when generating a bunch of sentences from grammar.gr and then seeing how many different parses they have. The short sentences that are generated by grammar.gr usually tend to have a lot lower parses
then the really big sentences(more complex). Usually for the short sentences they will tend to have 1 or 2 number of parses. For example the sentence: is it true that the sandwich wanted a president ? (Produces number of parses = 1) The difference is a lot when it comes to the really big sentences. For example a sentence like: is it true that every president with the pickle kissed every president in every delicious sandwich with every chief of staff with a pickled chief of staff on the floor on every sandwich in every sandwich with a president in every president with the president with the chief of staff with a president with a sandwich with the chief of staff with every floor with the chief of staff under every floor on the chief of staff on the president in the pickled sandwich with a floor ? (Produces number of parses = 6564120420)

grammar3.gr:
We see the same kind of patterns that we saw in grammar.gr. The shorter sentences tend to have a lower amount of parses compared to the longer sentences that are more complex. For example for a sentence like: the perplexed president pickled the desk ! (produced number of parses = 1). However, for a sentence that is larger and more complex like: that every chief of staff worked on that proposal on that chief of staff under a chief of staff under the president with it sighed under the desk and a chief of staff . (produced number of parses = 38). We do see however that the large gap between the number of parses between smaller and larger sentences has definitely closed
and this makes sense because in grammar3.gr we have rules that don't give us those really large and complex sentences.

5.
a)
- Why is p(best parse) so small? What probabilities were multiplied together to get its value of 5.144e-05? (Hint: Look at grammar.gr.)

  the value of p(best parse) is small because in practice, the parser is multiplying small fractional probabilities assigned internally by the system when computing the likelihood of each step.

- p(sentence) is the probability that randsent would generate this sentence. Why is it equal to p(best parse)?

p(sentence) is equal to p(best parse) because the best parse is the only possible derivation for the sentence in this grammar, with no alternative parses. Thus, the probability of the sentence is fully determined by that single parse.

- Why is the third number 1?
  Given the sentence, the probability that the best parse is the correct one is 100%. Since the grammar provides only one valid parse for this sentence, there is no ambiguity, and the best parse is guaranteed to be the correct one. So that's why the conditional probability is 1

b) What does it mean that the third number is 0.5 in this case? Why would it be exactly 0.5? (Hint:
Again, look at grammar.gr.)

The third number, p(best_parse | sentence) = 0.5, means that the parser found two equally probable parses for the second sentence, but it selected the best one (based on probabilities). It's exactly 0.5 because if there are multiple parses for a sentence, the parser needs to choose one. In this case, the parser found two valid derivations, and since they both had equal probabilities, the probability that the parser chooses any one of them is 0.5

c) After reading the whole 18-word corpus (including punctuation), the parser reports how well the grammar did at predicting the corpus. Explain exactly how the following numbers were calculated from the numbers above:

Cross-entropy is the average negative log probability per word. So first we take the log probability of the first sentence = log(5.144e-05) = -14.266 and then the second sentence which is log(6.202e-10) = -29.567. Now we just add these two together and get -43.833. Then we do - (-43.833) / 18 = 2.435 bits per word

d) Based on the above numbers, what perplexity per word did the grammar achieve on this corpus?
(Remember from lecture that perplexity is just a variation on cross-entropy.)

Perplexity = 2^H. So then we just do 2^(2.435) = 5.41. The grammar achieved a perplexity of 5.41 per word on this corpus

e) But the compression program might not be able to compress the following corpus too well.

Why not? What cross-entropy does the grammar achieve this time? Try it and explain

The compression program wouldn't be able to compress the following corpus well because the second sentence, "the president ate", is incomplete according to the grammar and hence is unparsable. For this corpus, the cross-entropy would be undefined or infinite due to the inability to parse the second sentence. (logarithms of 0 are undefined)

a) Commands:

```
  python3 randsent.py -g grammar2.gr -n 1000 > corpus.txt
  ./parse -P -g grammar2.gr < corpus.txt > parsed_output.txt
  grep "P(best_parse) =" parsed_output.txt | awk '{ print $4 }' | awk '{ sum += log($1);
count++ } END { print -sum/count/log(2) }'
```

The result was 26.1796 bits per word, indicating high entropy.
This means the grammar is quite "creative," generating a wide variety of sentences.
High entropy reflects a grammar that doesn't produce repetitive, predictable patterns, but instead generates diverse and unique sentence structures.

b)
The result for grammar3 I got was 34.852. The entropy of grammar3 (34.852 bits per word) is higher than the entropy of grammar2 (26.1796 bits per word).
This means that grammar3 is more "creative" or diverse in the sentences it generates, producing a wider variety of word sequences.
In contrast, grammar2 is less diverse and tends to generate more predictable sentences, which results in lower entropy.

c)
The attempt to compute entropy for the original grammar fails because the grammar is too constrained meaning that it's too simplistic this occurs because
of the lack of rules compared to what grammar2 and grammar3 have. This results
in zero probabilities for some sentences, which makes entropy calculations invalid.

7. If you generate a corpus from grammar2, then grammar2 should on average predict this corpus
better than grammar or grammar3 would. In other words, the entropy will be lower than the crossentropies. Check whether this is true: compute the numbers and discuss.

We already know for a corpus from grammar2 its entropy is 26.1796. However when I compute
cross-entropy of grammar2 with grammar3 I get 34.8716 and then when I compute
the cross entropy of grammar 2 with the original grammr I get 27.131. So we see

that grammar2 predicts its own corpus with lower entropy than grammar3 or the original grammar.
This means that it is better at predicting the sentences it generates compared to grammar3 or the original grammar.
This makes sense because grammar2 was designed to generate that particular set of sentences, so it should assign
higher probabilities to them compared to grammar3 or the original grammar.

PART 4:
Implemented yes-no questions to our grammar. This is how we did it:

We updated the ROOT symbol to handle questions as part of the possible structures it can generate. We did this by adding the line: 1 ROOT Q ? The next thing we did was actually add the rule for a Question(Q) we did this by adding the line:
1 Q Aux NP VP. We did this because because in English yes-no questions
they often begin with an auxiliary verb, then it's followed by the subject
this case it's the noun phrase and then the actual main verb and its objects/components
in this case the verb phrase. Another thing we did was add the present tense for
every single verb whether it's a TV or IV. For example worked and thought
were added as work and think. Then we ran into an issue when generating sentences,
the problem was that it would combine present tense verbs and past tense verbs
in the yes-no questions which would then make the questions grammatically incorrect, or for normal statements it would use present tense verbs and it would be grammatically incorrect. So what we did is separate the verb phrases into present and past tense verbs. We made the questions only deal with present tense verb phrases, and then everything we had before we set as past verb phrases.

Example output we have for the new grammar4 :
will Sally kiss and kiss the president ?
did the desk work ?
will a proposal and a president and the proposal kiss the pickle ?
will the desk perplex that president ?
will every sandwich kiss the president in it and understand Sally and the pickle on the floor under the floor ?
did the pickled chief of staff sigh on a desk ?
did every floor understand every sandwich ?

Implemented Wh-word questions/interrogatives:
We first added all of the wh-words and then created new question structures that handled the wh-words. We ran into a problem early on where some of the wh-words were tense sensitive, so we had to separate and handle them differently. Once we

handled this, it just came down to figuring out all the ways that specific wh-words interact with sentences: ex: how "who" interacts with an aux, if no aux then its a past verb tense, if it has an aux, then it's going to be a present tense verb. This phenomena was a lot easier to do because we also completed part b.

Example output we have for the updated grammar4 :

```
who sighed ?
who will kiss the proposal and kiss the pickle ?
where will it and the desk want the chief of staff ?
where did every desk on the sandwich sigh ?
what pickled and understood a president ?
what will the proposal perplex Sally on Sally on it in a sandwich ?
when did that floor understand that it sighed and sighed wanted it ?
when will the desk under Sally kiss Sally under it under that floor in a desk on it ?
why did a desk work and work ?
why did a president pickle and perplex Sally ?
```