```
Completed with errors. Exit status: 256
```

Out[2]: 256

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call dri
ve.mount("/content/drive", force_remount=True).
cp: cannot open '/content/drive/MyDrive/nlp_lab/project3/requirements.gdoc' for r
eading: Operation not supported
Requirement already satisfied: nltk==3.5 in /usr/local/lib/python3.11/dist-packag
es (from -r /content/drive/MyDrive/nlp_lab/project3/requirements.txt (line 1))
(3.5)
Requirement already satisfied: wget in /usr/local/lib/python3.11/dist-packages (f
rom -r /content/drive/MyDrive/nlp_lab/project3/requirements.txt (line 2)) (3.2)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages
(from nltk==3.5->-r /content/drive/MyDrive/nlp_lab/project3/requirements.txt (lin
e 1)) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages
(from nltk==3.5->-r /content/drive/MyDrive/nlp_lab/project3/requirements.txt (lin
e 1)) (1.4.2)
Requirement already satisfied: regex in /usr/local/lib/python3.11/dist-packages
(from nltk==3.5->-r /content/drive/MyDrive/nlp_lab/project3/requirements.txt (lin
e 1)) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (f
rom nltk==3.5->-r /content/drive/MyDrive/nlp_lab/project3/requirements.txt (line
1)) (4.67.1)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-3-0fdba98e8b60> in <cell line: 0>()
      1 # Initialize Otter
----> 2 import otter
      3 grader = otter.Notebook()

ModuleNotFoundError: No module named 'otter'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

%%latex \newcommand{\vect}[1]{\mathbf{#1}} \newcommand{\cnt}[1]{\sharp(#1)} \newcommand{\argmax}[1]{\underset{#1}{\operatorname{argmax}}} \newcommand{\softmax}{\operatorname{softmax}} \newcommand{\Prob}{\Pr} \newcommand{\given}{\,|\,}

Course 236299

Project 3: Parsing – The CKY Algorithm

Constituency parsing is the recovery of a labeled hierarchical structure, a *parse tree* for a sentence of a natural language. It is a core intermediary task in natural-language processing, as the meanings of sentences are related to their structure.

In this project, you will implement the CKY algorithm for parsing strings relative to context-free grammars (CFG). You will implement versions for both non-probabilistic context-free grammars (CFG) and probabilistic grammars (PCFG) and apply them to the parsing of ATIS queries.

The project is structured into five parts:

1. Finish a CFG for the ATIS dataset.
2. Implement the CKY algorithm for *recognizing* grammatical sentences, that is, determining whether a parse exists for a given sentence.
3. Extend the CKY algorithm for *parsing* sentences, that is, constructing the parse trees for a given sentence.
4. Construct a probabilistic context-free grammar (PCFG) based on a CFG.
5. Extend the CKY algorithm to PCFGs, allowing the construction of the most probable parse tree for a sentence according to a PCFG.

Setup

```
Downloading train.nl from https://raw.githubusercontent.com/nlp-course/data/master/ATIS/

Downloading train.trees from https://raw.githubusercontent.com/nlp-course/data/master/ATIS/

Downloading test.trees from https://raw.githubusercontent.com/nlp-course/data/master/ATIS/

Downloading evalb.py from https://raw.githubusercontent.com/nlp-course/data/master/scripts/trees/

Downloading transform.py from https://raw.githubusercontent.com/nlp-course/data/master/scripts/trees/

Downloading tree.py from https://raw.githubusercontent.com/nlp-course/data/master/scripts/trees/
```

A custom ATIS grammar

To parse, we need a grammar. In this project, you will use a hand-crafted grammar for a fragment of the ATIS dataset. The grammar is written in a "semantic grammar" style, in which the nonterminals tend to correspond to semantic classes of phrases, rather than syntactic classes. By using this style, we can more closely tune the grammar to the application, though we lose generality and transferability to other applications. The

grammar will be used again in the next project segment for a question-answering application.

We download the grammar to make it available.

```
Downloading grammar_distrib3 from https://raw.githubusercontent.com/nlp-course/da
ta/master/ATIS/
```

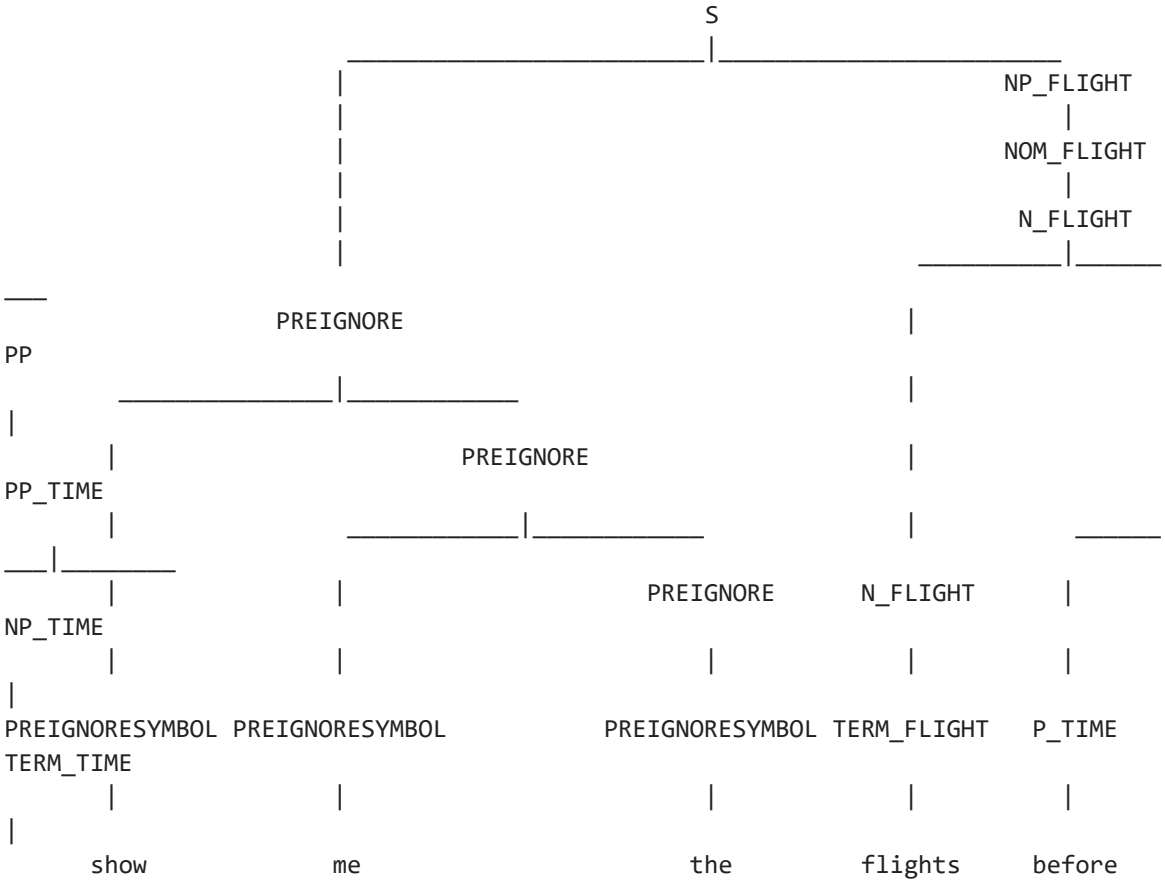
Take a look at the file `data/grammar_distrib3` that you've just downloaded. The grammar is written in a format that extends the NLTK format expected by `CFG.fromstring`. We've provided functions to make use of this format in the file `scripts/transform.py`. You should familiarize yourself with this format by checking out the documentation in that file.

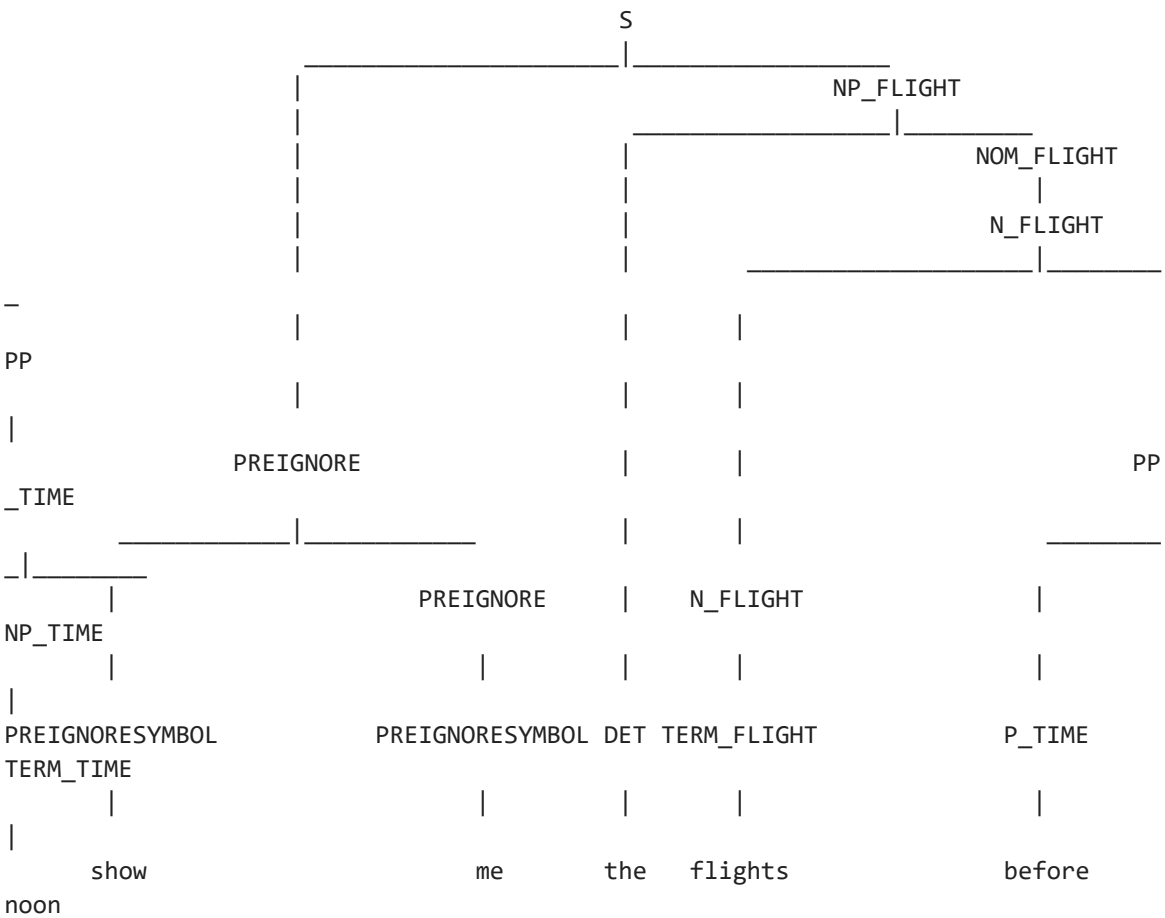
> We made a copy of this grammar for you as `data/grammar`. This is the file you'll be modifying in the next section. You can leave it alone for now.

As described there, we can read the grammar in and convert it into NLTK's grammar format using the provided `xform.read_augmented_grammar` function.

To verify that the ATIS grammar that we distributed is working, we can parse a sentence using a built-in NLTK parser. We'll use a tokenizer built with NLTK's tokenizing apparatus.

```
['are', 'there', 'any', 'first-class', 'flights', 'at', '11', 'pm', 'for', 'les
s', 'than', '$3.50', '?']
```





Testing the coverage of the grammar

We can get a sense of how well the grammar covers the ATIS query language by measuring the proportion of queries in the training set that are parsable by the grammar. We define a `coverage` function to carry out this evaluation.

> Warning: It may take a long time to parse all of the sentence in the training corpus, on the order of 30 minutes. You may want to start with just the first few sentences in the corpus. The `coverage` function below makes it easy to do so, and in the code below we just test coverage on the first 50 sentences.

```
100%|██████████| 50/50 [00:00<00:00, 585.74it/s]
```

Out[20]:
```
0.0
```

Sadly, you'll find that the coverage of the grammar is extraordinarily poor. That's because it is missing crucial parts of the grammar, especially phrases about *places*, which play a role in essentially every ATIS query. You'll need to complete the grammar before it can be useful.

Part 1: Finish the CFG for the ATIS dataset

Consider the following query:

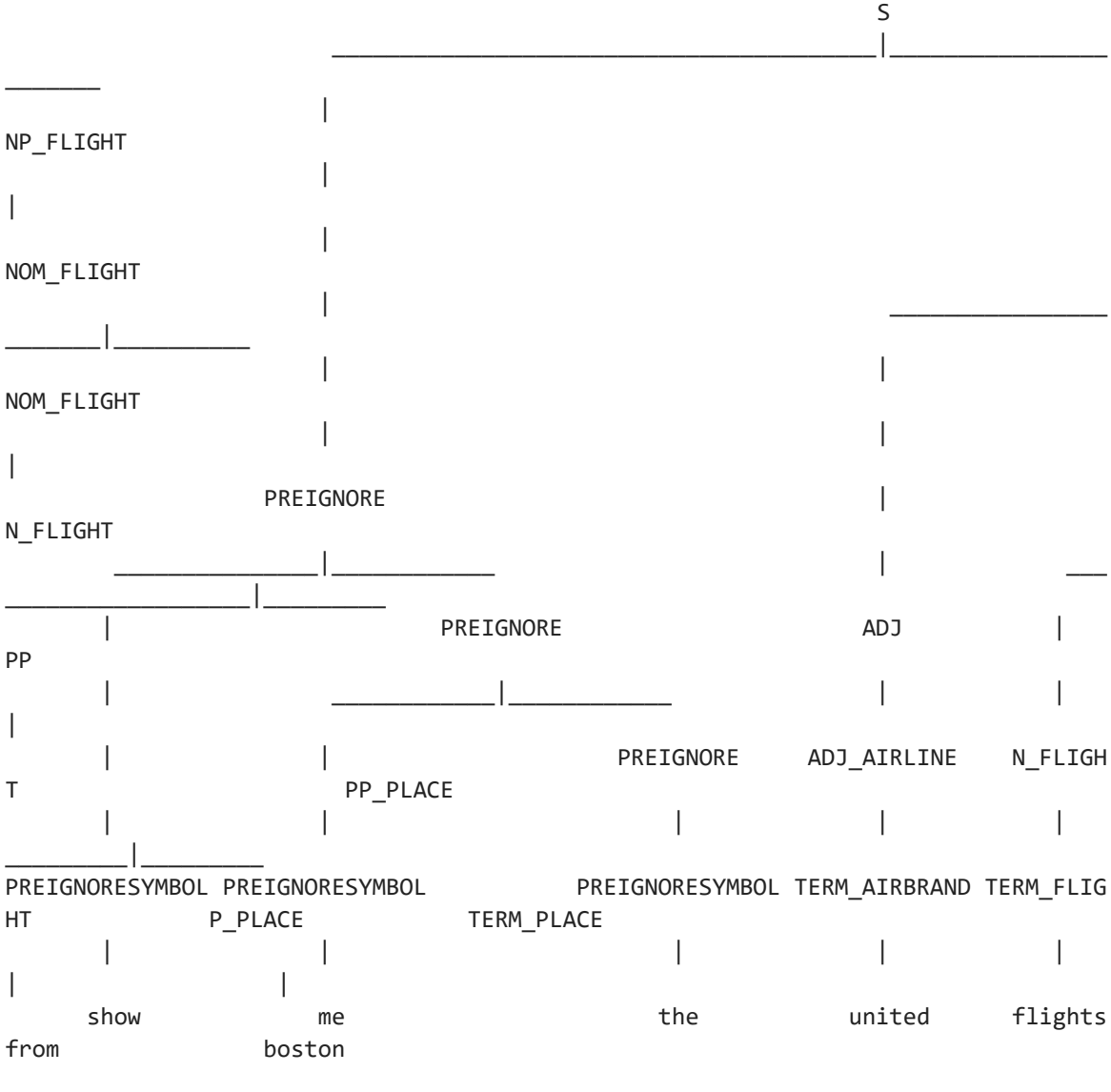
You'll notice that the grammar we distributed doesn't handle this query because it doesn't have a subgrammar for airline information (`"united"`) or for places (`"from boston"`).

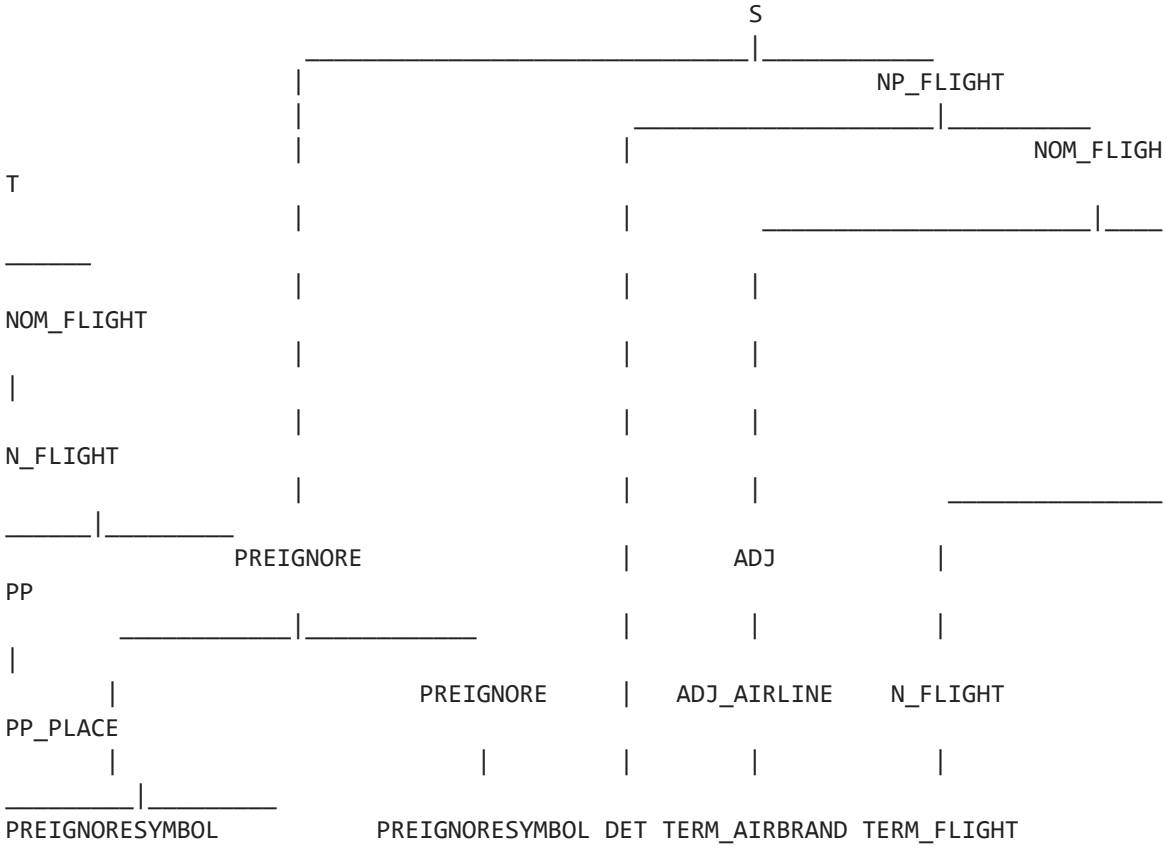
Out[22]:
```
0
```

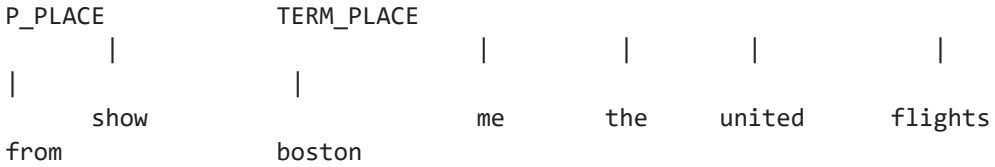
Follow the instructions in the grammar file `data/grammar` to add further coverage to the grammar. (You can and should leave the `data/grammar_distrib3` copy alone and use it for reference.)

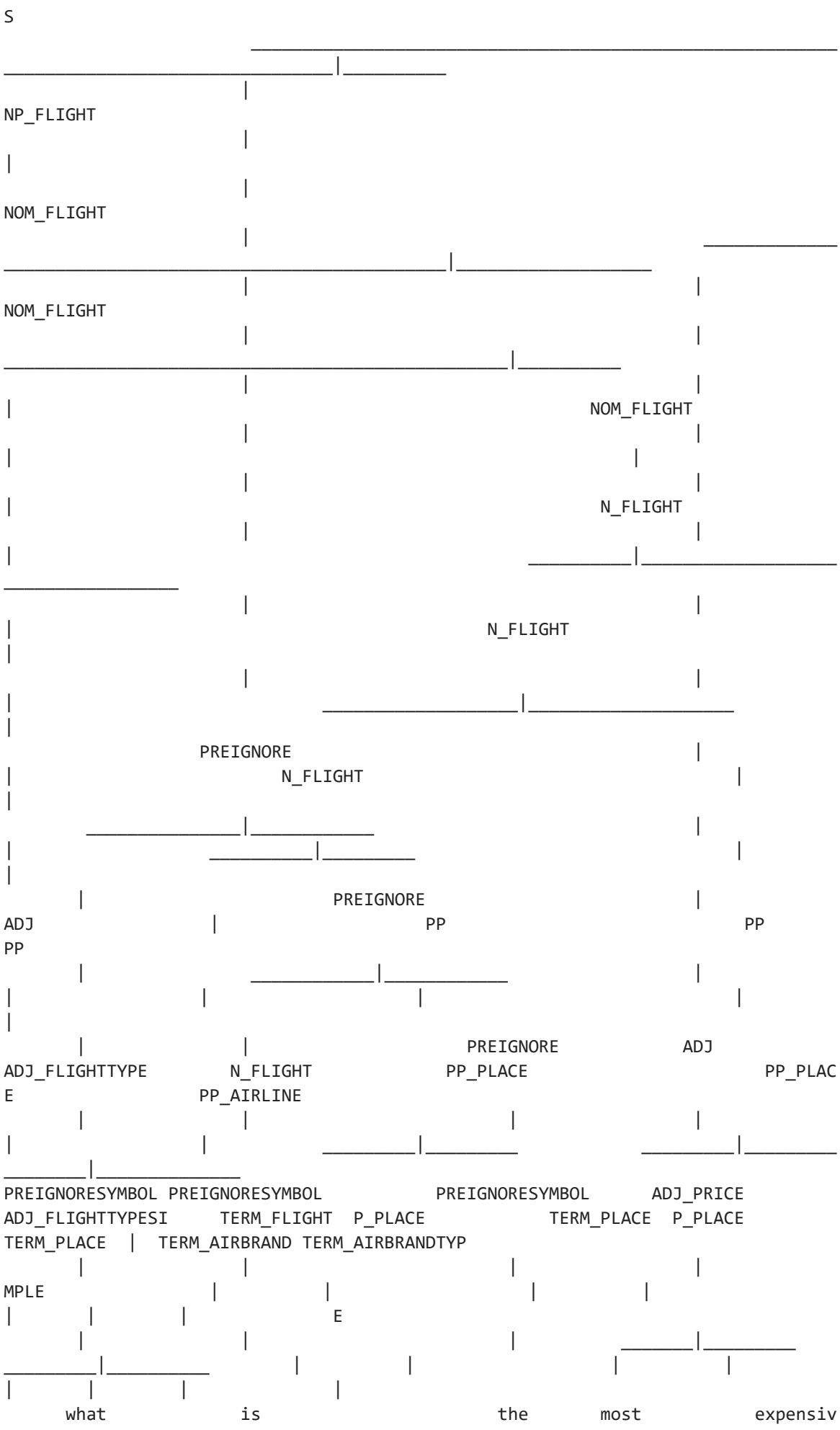
We'll define a parser based on your modified grammar, so we can compare it against the distributed grammar. Once you've modified the grammar, this test sentence should have at least one parse.

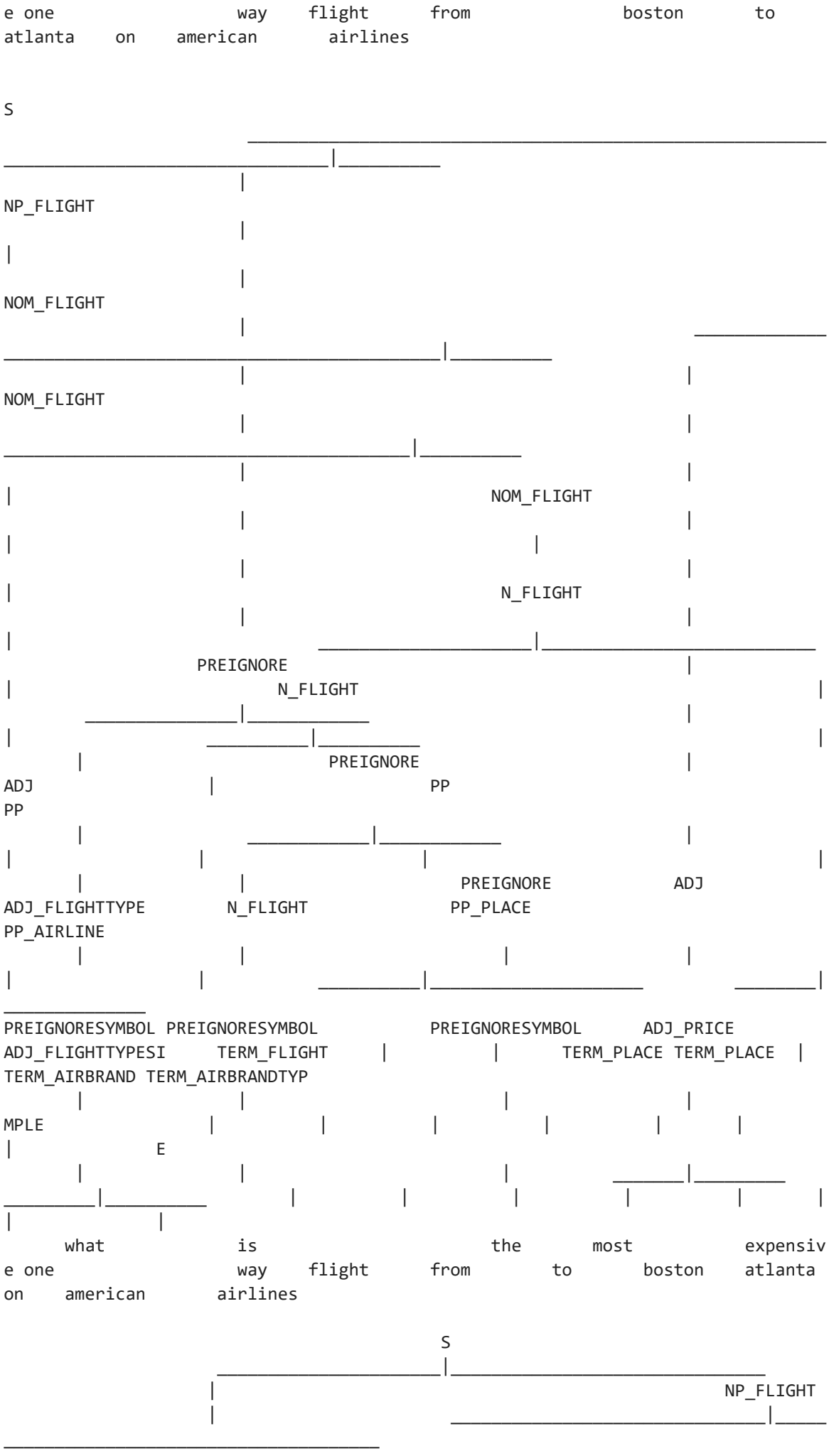
> **Hint:** You can search for "TODO" in `data/grammar` to find the two places to add grammar rules.

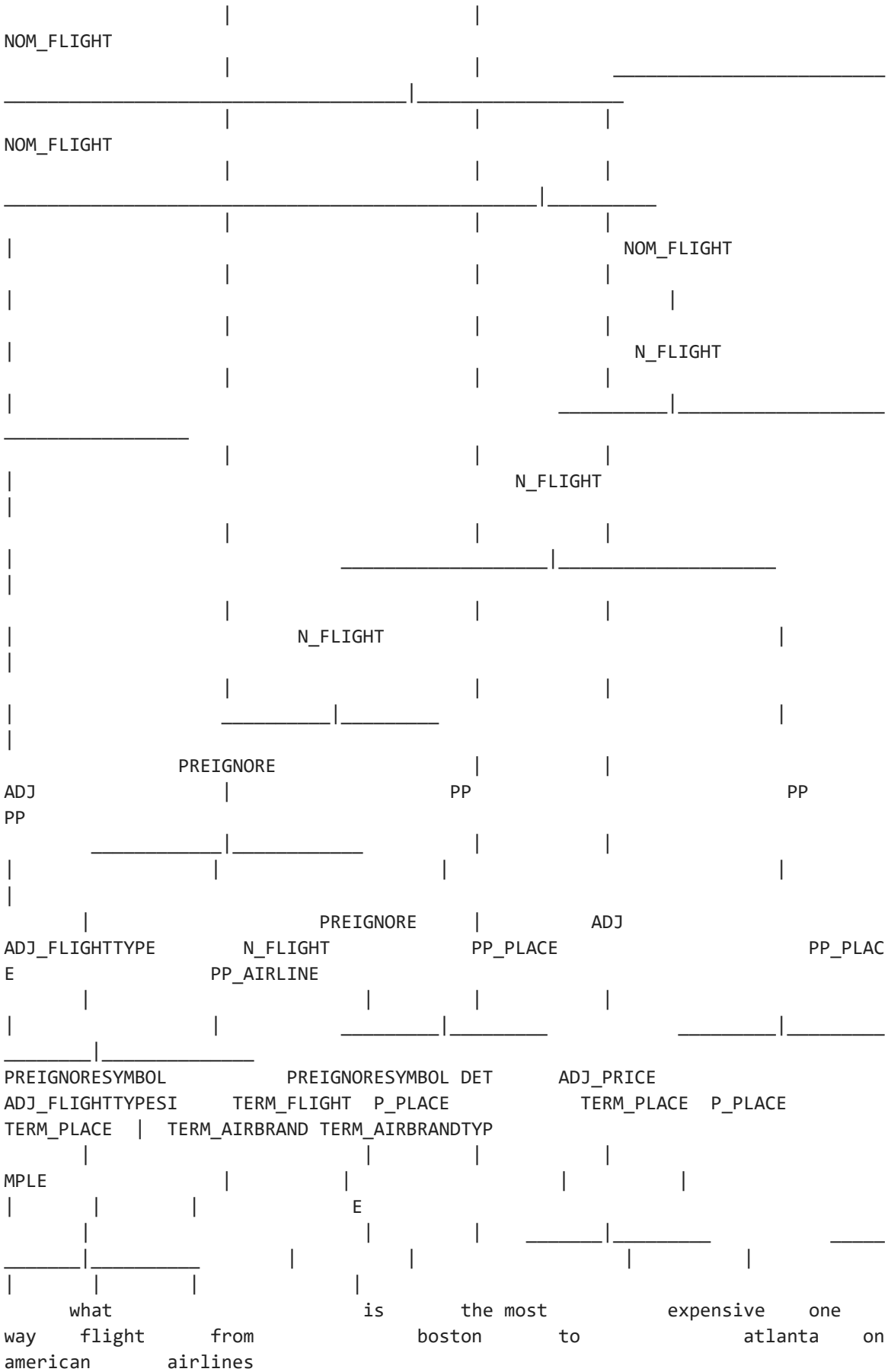

S
NP_FLIGHT
NOM_FLIGHT
NOM_FLIGHT
PREIGNORE
N_FLIGHT
PREIGNORE
ADJ
PP
PREIGNORE
ADJ_AIRLINE
N_FLIGHT
PP_PLACE
PREIGNORESYMBOL
PREIGNORESYMBOL
PREIGNORESYMBOL
TERM_AIRBRAND
TERM_FLIGHT
P_PLACE
TERM_PLACE
show
me
the
united
flights
from
boston

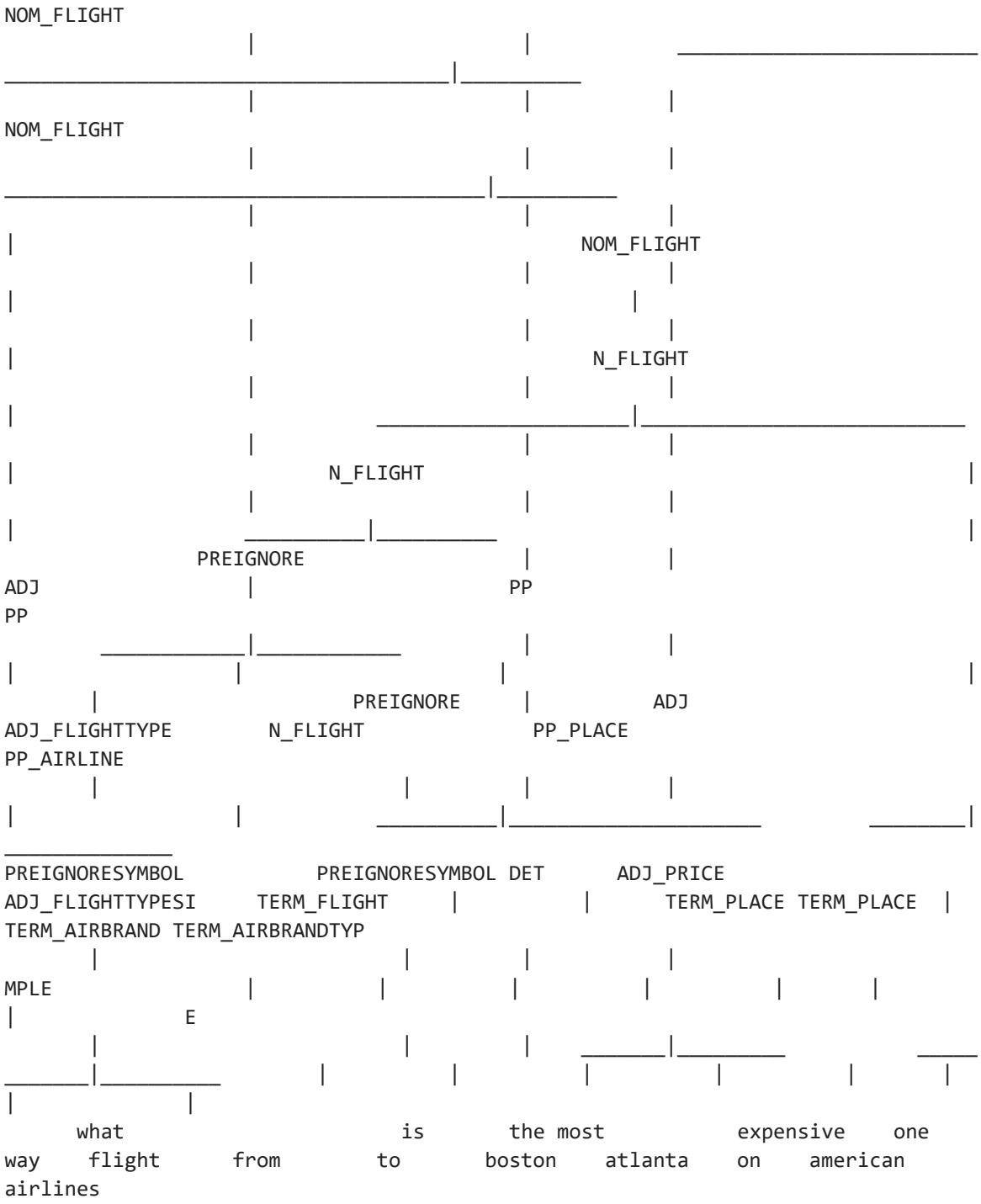

S
NP_FLIGHT
NOM_FLIGHT
NOM_FLIGHT
N_FLIGHT
PREIGNORE
ADJ
PP
PREIGNORE
ADJ_AIRLINE
N_FLIGHT
PP_PLACE
PREIGNORESYMBOL
PREIGNORESYMBOL
DET
TERM_AIRBRAND
TERM_FLIGHT


P_PLACE
TERM_PLACE
show
me
the
united
flights
from
boston


S
NP_FLIGHT
NOM_FLIGHT
NOM_FLIGHT
NOM_FLIGHT
N_FLIGHT
N_FLIGHT
PREIGNORE
N_FLIGHT
PREIGNORE
ADJ
PP
PP
PP
ADJ_FLIGHTTYPE
N_FLIGHT
PREIGNORE
PP_PLACE
ADJ
PP_PLACE
PP_AIRLINE
PREIGNORESYMBOL
PREIGNORESYMBOL
PREIGNORESYMBOL
ADJ_PRICE
ADJ_FLIGHTTYPESI
TERM_FLIGHT
P_PLACE
TERM_PLACE
P_PLACE
TERM_PLACE
TERM_AIRBRAND
TERM_AIRBRANDTYP
MPLE
E
what
is
the
most
expensiv


e one way flight from boston to
atlanta on american airlines
S
NP_FLIGHT
NOM_FLIGHT
NOM_FLIGHT
NOM_FLIGHT
N_FLIGHT
PREIGNORE
N_FLIGHT
PREIGNORE
ADJ
PP
PREIGNORE
ADJ
ADJ_FLIGHTTYPE
N_FLIGHT
PP_PLACE
PP_AIRLINE
PREIGNORESYMBOL PREIGNORESYMBOL
PREIGNORESYMBOL
ADJ_PRICE
ADJ_FLIGHTTYPESI
TERM_FLIGHT
TERM_PLACE TERM_PLACE
TERM_AIRBRAND TERM_AIRBRANDTYP
what is the most expensiv
e one way flight from to boston atlanta
on american airlines
S
NP_FLIGHT


NOM_FLIGHT
NOM_FLIGHT
NOM_FLIGHT
N_FLIGHT
N_FLIGHT
N_FLIGHT
N_FLIGHT
PREIGNORE
ADJ
PP
PP
PREIGNORE
ADJ
ADJ_FLIGHTTYPE
N_FLIGHT
PP_PLACE
PP_PLACE
PP_AIRLINE
PREIGNORESYMBOL
PREIGNORESYMBOL
DET
ADJ_PRICE
ADJ_FLIGHTTYPESI
TERM_FLIGHT
P_PLACE
TERM_PLACE
P_PLACE
TERM_PLACE
TERM_AIRBRAND
TERM_AIRBRANDTYP
what
is
the most
expensive
one way
flight
from
boston
to
atlanta
on american
airlines
S
NP_FLIGHT



Once you're done adding to the grammar, to check your grammar, we'll compute the grammar's coverage of the ATIS training corpus as before. **This grammar should be expected to cover about half of the sentences in the first 50 sentences, and a third of the entire training corpus.**

```
100%|██████████| 50/50 [00:00<00:00, 316.96it/s]
```

Out[25]: 0.48

CFG recognition via the CKY algorithm

Now we turn to implementing recognizers and parsers using the CKY algorithm. We start with a recognizer, which should return `True` or `False` if a grammar does or does not admit a sentence as grammatical.

Converting the grammar to CNF for use by the CKY algorithm

The CKY algorithm requires the grammar to be in Chomsky normal form (CNF). That is, only rules of the forms

$$A \to B\,C$$
$$A \to a$$

are allowed, where A, B, C are nonterminals and a is a terminal symbol.

However, in some downstream applications (such as the next project segment) we want to use grammar rules of more general forms, such as $A \to B\,C\,D$. Indeed, the ATIS grammar you've been working on makes use of the additional expressivity beyond CNF.

To satisfy both of these constraints, we will convert the grammar to CNF, parse using CKY, and then convert the returned parse trees back to the form of the original grammar. We provide some useful functions for performing these transformations in the file `scripts/transform.py`, already loaded above and imported as `xform`.

To convert a grammar to CNF:

```
cnf_grammar, cnf_grammar_wunaries = xform.get_cnf_grammar(grammar)
```

To convert a tree output from CKY back to the original form of the grammar:

```
xform.un_cnf(tree, cnf_grammar_wunaries)
```

> We pass into `un_cnf` a version of the grammar before removing unary nonterminal productions, `cnf_grammar_wunaries`. The `cnf_grammar_wunaries` is returned as the second part of the returned value of `get_cnf_grammar` for just this purpose.

In the next sections, you'll write your own recognizers and parsers based on the CKY algorithm that can operate on this grammar.

Part 2: Implement a CKY recognizer

Implement a *recognizer* using the CKY algorithm to determine if a sentence `tokens` is parsable according to a `grammar`. The labs and J&M Chapter 13, both of which provide appropriate pseudo-code for CKY, should be useful references here.

> **Hint:** Recall that you can get the production rules of a grammar using `grammar.productions()`.

> Throughout this project segment, you should use `grammar.start()` to get the special start symbol from the grammar instead of using `S`, since some grammars may use a different start symbol, such as `TOP`.

You can test your recognizer on a few examples, both grammatical and ungrammatical, as below.

```
+    flights from boston
+    show me flights from boston
+    show me united flights before noon
+    are there any twa flights available tomorrow
-    show me flights united are there any
```

You can also verify that the CKY recognizer has the same coverage as the NLTK parser.

```
100%|██████████| 50/50 [00:08<00:00,  5.71it/s]
```

Out[29]: 0.48

Part 3: Implement a CKY parser

In part 2, you implemented a context-free grammar *recognizer*. Next, you'll implement a *parser*.

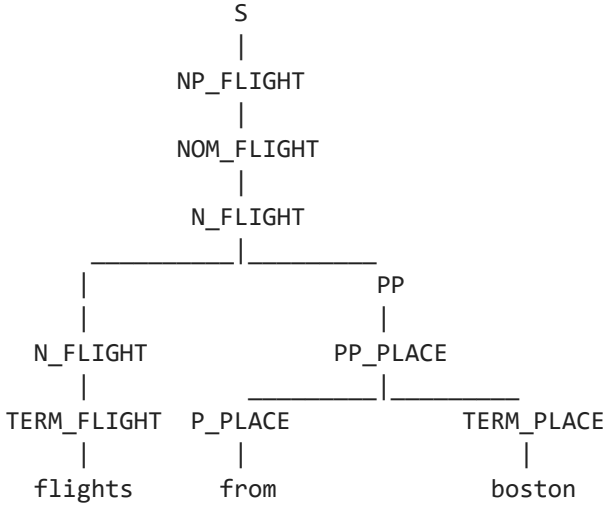
Implement the CKY algorithm for parsing with CFGs as a function `cky_parse`, which takes a grammar and a list of tokens and returns a single parse of the tokens as specified by the grammar, or `None` if there are no parses. You should only need to add a few lines of code to your CKY recognizer to achieve this, to implement the necessary back-pointers. The function should return an NLTK tree, which you will need to reconstruct from the back pointers table.

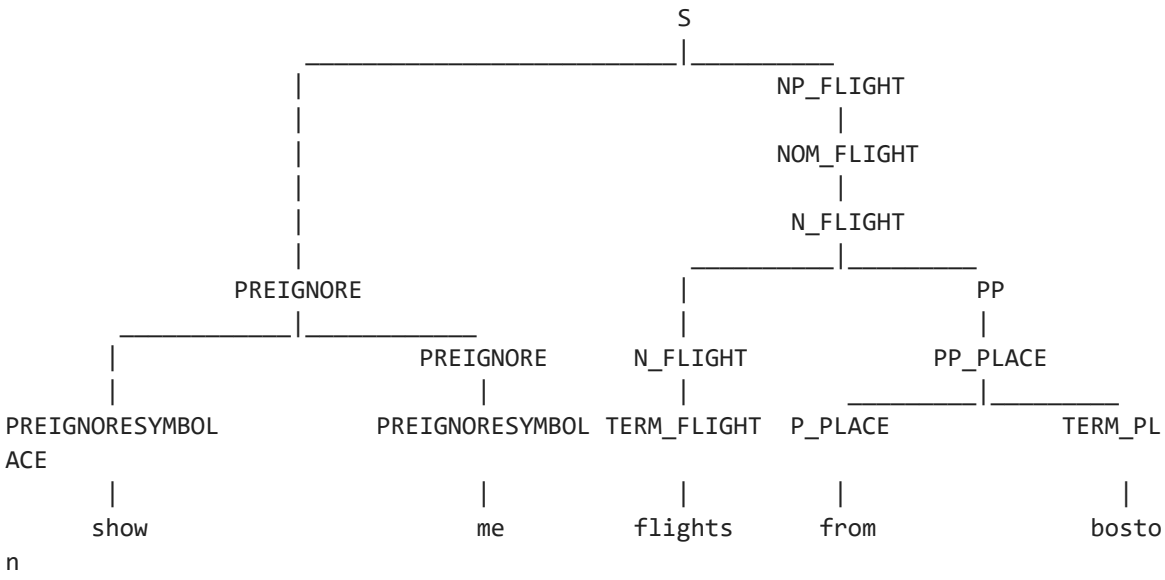
> **Hint:** You can build NLTK trees directly using `Tree`, which takes a string for the root nonterminal and, recursively, a list of children.
>
> Alternatively, you can construct a tree using `Tree.fromstring`. A tree string will look like this example:
>
> `"(S (A B) (C (D E) (F G)))"`
> which corresponds to the following tree (drawn using `tree.pretty_print()`):
>
> ```
>       S
>    ___|___
>   |       C
>   |    ___|___
>   A   D       F
>   |   |       |
>   B   E       G
> ```

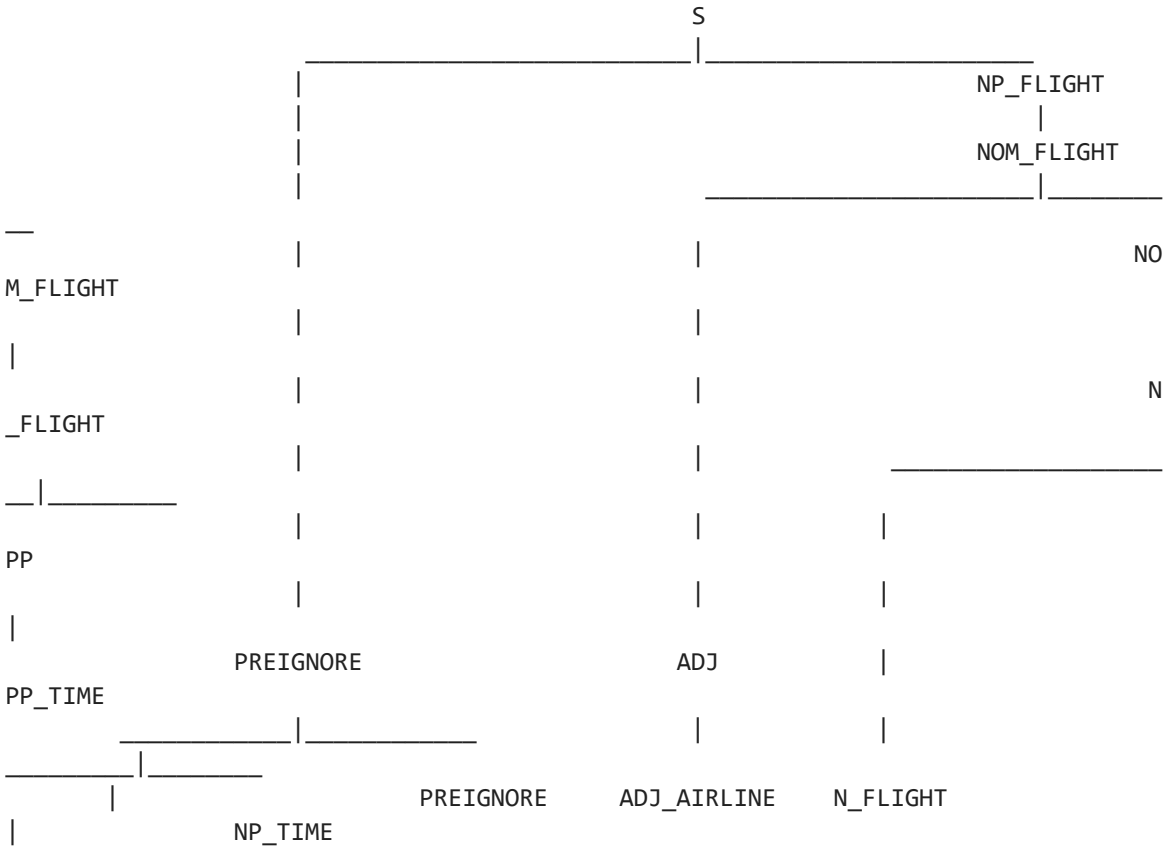
> **Hint:** You may need to extract from a `Nonterminal` its corresponding string. The `Nonterminal.__str__` method or f-string

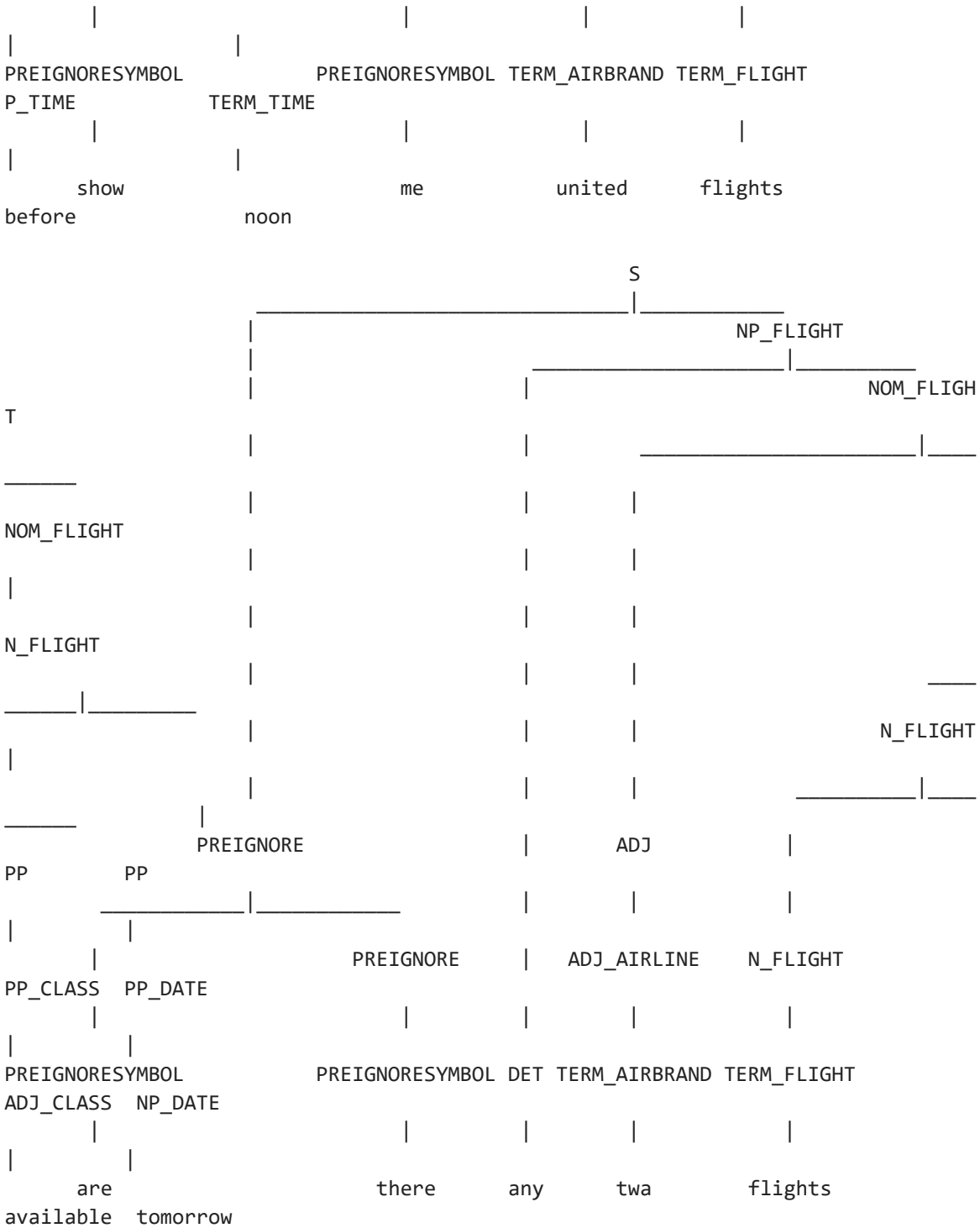
> `f'{Nonterminal}'` accomplishes this.

You can test your code on the test sentences provided above:


S
NP_FLIGHT
NOM_FLIGHT
N_FLIGHT
PP
N_FLIGHT
PP_PLACE
TERM_FLIGHT
P_PLACE
TERM_PLACE
flights
from
boston


S
NP_FLIGHT
NOM_FLIGHT
N_FLIGHT
PREIGNORE
PP
PREIGNORE
N_FLIGHT
PP_PLACE
PREIGNORESYMBOL
PREIGNORESYMBOL
TERM_FLIGHT
P_PLACE
TERM_PLACE
show
me
flights
from
boston


S
NP_FLIGHT
NOM_FLIGHT
NOM_FLIGHT
N_FLIGHT
PP
PREIGNORE
ADJ
PP_TIME
PREIGNORE
ADJ_AIRLINE
N_FLIGHT
NP_TIME



failed to parse: show me flights united are there any

You can also compare against the built-in NLTK parser that we constructed above:

```
Reference parses:
(S
  (NP_FLIGHT
    (NOM_FLIGHT
      (N_FLIGHT
        (N_FLIGHT (TERM_FLIGHT flights))
        (PP (PP_PLACE (P_PLACE from) (TERM_PLACE boston)))))))

Predicted parse:
(S
  (NP_FLIGHT
    (NOM_FLIGHT
      (N_FLIGHT
        (N_FLIGHT (TERM_FLIGHT flights))
        (PP (PP_PLACE (P_PLACE from) (TERM_PLACE boston)))))))

SUCCESS!
Reference parses:
(S
  (PREIGNORE (PREIGNORESYMBOL show) (PREIGNORE (PREIGNORESYMBOL me)))
  (NP_FLIGHT
    (NOM_FLIGHT
      (N_FLIGHT
        (N_FLIGHT (TERM_FLIGHT flights))
        (PP (PP_PLACE (P_PLACE from) (TERM_PLACE boston)))))))

Predicted parse:
(S
  (PREIGNORE (PREIGNORESYMBOL show) (PREIGNORE (PREIGNORESYMBOL me)))
  (NP_FLIGHT
    (NOM_FLIGHT
      (N_FLIGHT
        (N_FLIGHT (TERM_FLIGHT flights))
        (PP (PP_PLACE (P_PLACE from) (TERM_PLACE boston)))))))

SUCCESS!
Reference parses:
(S
  (PREIGNORE (PREIGNORESYMBOL show) (PREIGNORE (PREIGNORESYMBOL me)))
  (NP_FLIGHT
    (NOM_FLIGHT
      (ADJ (ADJ_AIRLINE (TERM_AIRBRAND united)))
      (NOM_FLIGHT
        (N_FLIGHT
          (N_FLIGHT (TERM_FLIGHT flights))
          (PP (PP_TIME (P_TIME before) (NP_TIME (TERM_TIME noon)))))))))

Predicted parse:
(S
  (PREIGNORE (PREIGNORESYMBOL show) (PREIGNORE (PREIGNORESYMBOL me)))
  (NP_FLIGHT
    (NOM_FLIGHT
      (ADJ (ADJ_AIRLINE (TERM_AIRBRAND united)))
      (NOM_FLIGHT
        (N_FLIGHT
          (N_FLIGHT (TERM_FLIGHT flights))
          (PP (PP_TIME (P_TIME before) (NP_TIME (TERM_TIME noon)))))))))

SUCCESS!
```

```
Reference parses:
(S
  (PREIGNORE
    (PREIGNORESYMBOL are)
    (PREIGNORE (PREIGNORESYMBOL there)))
  (NP_FLIGHT
    (DET any)
    (NOM_FLIGHT
      (ADJ (ADJ_AIRLINE (TERM_AIRBRAND twa)))
      (NOM_FLIGHT
        (N_FLIGHT
          (N_FLIGHT
            (N_FLIGHT (TERM_FLIGHT flights))
            (PP (PP_CLASS (ADJ_CLASS available))))
          (PP (PP_DATE (NP_DATE tomorrow))))))))

Predicted parse:
(S
  (PREIGNORE
    (PREIGNORESYMBOL are)
    (PREIGNORE (PREIGNORESYMBOL there)))
  (NP_FLIGHT
    (DET any)
    (NOM_FLIGHT
      (ADJ (ADJ_AIRLINE (TERM_AIRBRAND twa)))
      (NOM_FLIGHT
        (N_FLIGHT
          (N_FLIGHT
            (N_FLIGHT (TERM_FLIGHT flights))
            (PP (PP_CLASS (ADJ_CLASS available))))
          (PP (PP_DATE (NP_DATE tomorrow))))))))

SUCCESS!
Reference parses:

Predicted parse:
None

SUCCESS!
```

Again, we test the coverage as a way of verifying that your parser works consistently with the recognizer and the NLTK parser.

```
100%|██████████| 50/50 [00:06<00:00,  7.61it/s]
```

Out[33]: 0.48

Probabilistic CFG parsing via the CKY algorithm

In practice, we want to work with grammars that cover nearly all the language we expect to come across for a given application. This leads to an explosion of rules and a large number of possible parses for any one sentence. To remove ambiguity between the different parses, it's desirable to move to probabilistic context-free grammars (PCFG). In this part of the assignment, you will construct a PCFG from training data, parse using a

probabilistic version of CKY, and evaluate the quality of the resulting parses against gold trees.

Part 4: PCFG construction

Compared to CFGs, PCFGs need to assign probabilities to grammar rules. For this goal, you'll write a function `pcfg_from_trees` that takes a list of strings describing a corpus of trees and returns an NLTK PCFG trained on that set of trees.

> We expect you to implement `pcfg_from_trees` directly. You should **not** use the `induce_pcfg` function in implementing your solution.

We want the PCFG to be in CNF format because the probabilistic version of CKY that you'll implement next also requires the grammar to be in CNF. However, the gold trees are not in CNF form, so in this case you will need to convert the gold *trees* to CNF before building the PCFG from them. To accomplish this, you should use the `treetransforms` package from `nltk`, which includes functions for converting to and from CNF. In particular, you'll want to make use of `treetransforms.collapse_unary` followed by `treetransforms.chomsky_normal_form` to convert a tree to its binarized version. You can then get the counts for all of the productions used in the trees, and then normalize them to probabilities so that the probabilities of all rules with the same left-hand side sum to 1.

We'll use the `pcfg_from_trees` function that you define later for parsing.

> To convert an `nltk.Tree` object `t` to CNF, you can use the below code. Note that it's different from the `xform` functions we used before as we are converting *trees*, not *grammars*.
>
> ```
> treetransforms.collapse_unary(t, collapsePOS=True)
> treetransforms.chomsky_normal_form(t) # After this the tree
> will be in CNF
> ```

> To construct a PCFG production, see nltk.grammar.ProbabilisticProduction.

> To construct a PCFG with a given start state and set of productions, see `nltk.grammar.PCFG`.

We can now train a PCFG on the *train* split `train.trees` that we downloaded in the setup at the start of the notebook.

[TOP -> S+VP PUNC [0.208955], TOP -> SQ PUNC [0.0447761], TOP -> S PUNC [0.11513
9], TOP -> SBARQ PUNC [0.217484], TOP -> FRAG PUNC [0.0597015], TOP -> FRAG+NP PU
NC [0.223881], TOP -> INTJ+UH PUNC [0.00426439], TOP -> FRAG+NP+NN PUNC [0.004264
39], TOP -> X+SBARQ PUNC [0.0021322], TOP -> SBAR PUNC [0.0021322], TOP -> FRAG+V
P PUNC [0.0170576], TOP -> NP PUNC [0.00639659], TOP -> FRAG+NP+NNP PUNC [0.00213
22], TOP -> FRAG+WHNP PUNC [0.0703625], TOP -> FRAG+PP PUNC [0.010661], TOP -> FR
AG+ADJP+JJ PUNC [0.00426439], TOP -> ADJP+JJ PUNC [0.0021322], TOP -> FRAG+ADJP+J
JS PUNC [0.0021322], TOP -> X+S+VP PUNC [0.0021322], S+VP -> VB NP [0.122302], S+
VP -> VBP PP [0.0143885], S+VP -> TO VP [0.151079], S+VP -> MD VP [0.0143885], S+
VP -> VBZ S+VP|<PP-PP> [0.0143885], S+VP -> VB S+VP|<NP+PRP-NP> [0.52518], S+VP -
> VB S+VP|<PP-PP> [0.00719424], S+VP -> VB S+VP|<NP+PRP-PP> [0.00719424], S+VP ->
VB NP+PRP [0.0143885], S+VP -> VBP NP [0.0143885], S+VP -> VBZ PP [0.0215827], S+
VP -> VBP NP+NN [0.0143885], S+VP -> VB S+VP|<NP+PRP-NP-INTJ+UH> [0.0143885], S+V
P -> VBZ NP [0.00719424], S+VP -> VB NP+NNS [0.0143885], S+VP -> VBZ ADVP+RB [0.0
0719424], S+VP -> VBP ADVP [0.00719424], S+VP -> VP S+VP|<CC-VP> [0.0143885], S+V
P -> VBP S+VP|<RB-VP> [0.00719424], S+VP -> VBZ NP+NN [0.00719424], VB -> 'List'
[0.0180723], VB -> 'serve' [0.0421687], VB -> 'have' [0.0301205], VB -> 'like'
[0.10241], VB -> 'visit' [0.0120482], VB -> 'arrive' [0.0240964], VB -> 'return'
[0.0120482], VB -> 'leave' [0.0240964], VB -> 'find' [0.0120482], VB -> 'be' [0.0
120482], VB -> 'Show' [0.518072], VB -> 'go' [0.0120482], VB -> 'Tell' [0.006024
1], VB -> 'Thank' [0.0120482], VB -> 'Book' [0.0060241], VB -> 'travel' [0.018072
3], VB -> 'know' [0.0060241], VB -> 'stop' [0.0060241], VB -> 'fly' [0.0301205],
VB -> 'list' [0.0240964], VB -> 'make' [0.0060241], VB -> 'show' [0.0180723], VB
-> 'Give' [0.0060241], VB -> 'display' [0.0060241], VB -> 'get' [0.0060241], VB -
> 'cancel' [0.0060241], VB -> 'Explain' [0.0180723], VB -> 'Display' [0.0060241],
NP -> NP NP|<PP-PP-SBAR> [0.0057971], NP -> DT NNS [0.0855072], NP -> DT NN [0.10
1449], NP -> NP NP|<PP-VP-VP> [0.00144928], NP -> NP PP [0.0405797], NP -> NP SBA
R [0.0188406], NP -> CD NNS [0.00289855], NP -> NP+NNP NP [0.0130435], NP -> NNP
NNP [0.192754], NP -> NP+CD PP [0.00144928], NP -> DT JJ [0.00724638], NP -> NP N
P|<VP-SBAR> [0.00144928], NP -> NP NP|<PP-PP-PP> [0.0115942], NP -> NNP NP|<NNP-N
NP> [0.0376812], NP -> QP RB [0.00434783], NP -> NP NP|<PP-PP-NP> [0.00434783], N
P -> JJ NNP [0.00434783], NP -> DT NP|<NN-NN> [0.0115942], NP -> DT NP|<JJ-NN-NN>
[0.0057971], NP -> CD RB [0.0304348], NP -> DT NP|<NNP-NN> [0.00289855], NP -> NP
VP [0.0057971], NP -> PDT NP|<DT-NNS> [0.00434783], NP -> NP NP|<PP-PP> [0.049275
4], NP -> DT NP|<NN-NN-NN> [0.00289855], NP -> CD NP|<CC-CD-RB> [0.00289855], NP
-> DT NP|<JJ-NN> [0.0130435], NP -> DT NX [0.00144928], NP -> NP NP|<PP-PP-NP+NN>
[0.00144928], NP -> NP NP|<PP-PP-VP> [0.0101449], NP -> NN NN [0.0130435], NP ->
NP+NNP NP+NNP [0.00724638], NP -> DT NP|<NNP-NNP-CD-CD-NN> [0.00144928], NP -> NN
NP|<CD-CD-CD> [0.00289855], NP -> NNP JJ [0.00289855], NP -> NP NP|<NP+NN-PP-PP>
[0.00144928], NP -> NP NP|<NP+NN-PP-PP-PP> [0.00144928], NP -> DT NP|<JJS-NN> [0.
0101449], NP -> NP NP|<CC-NP> [0.0115942], NP -> NP NP|<NP-PP-X+TO-PP> [0.0014492
8], NP -> NP NP|<PP-PP-PP-NP+NN> [0.00144928], NP -> NP NP|<NN-CD-CD-CD> [0.00144
928], NP -> NNP POS [0.00289855], NP -> NP NP|<NP-PP-PP> [0.00144928], NP -> DT N
P|<NN-NNS> [0.00869565], NP -> DT NP|<CD-CD-NN> [0.00144928], NP -> DT NP|<CD-CD-
CD-NN> [0.00144928], NP -> CD NP|<CD-CD> [0.00434783], NP -> CD NP|<CD-CD-RB> [0.
00144928], NP -> DT NP|<JJ-NN-NNS> [0.00289855], NP -> NN NNS [0.0101449], NP ->
NP NP|<CC-ADVP+RB-NP> [0.00144928], NP -> NP+NN NP|<PP-PP> [0.0057971], NP -> NP+
NN PP [0.00144928], NP -> JJ NN [0.015942], NP -> NP+NN NP|<CC-NP> [0.00144928],
NP -> NNP NP|<CD-CD> [0.00144928], NP -> CD CD [0.00289855], NP -> NNP NN [0.0173
913], NP -> DT NP|<JJS-NN-NN> [0.00289855], NP -> CD NN [0.0144928], NP -> DT NP|
<NN-SYM-CD> [0.00144928], NP -> QP NN [0.00289855], NP -> NP NP|<SBAR-PP> [0.0014
4928], NP -> JJ NNS [0.00724638], NP -> NNP NP|<NNP-NN-CD-CD> [0.00289855], NP ->
NP+NNP NP|<CC-NP> [0.00869565], NP -> NP ADJP+JJ [0.00144928], NP -> NNP NP|<CD-C
D-CD> [0.00144928], NP -> DT NP|<JJ-X+JJ-NNS> [0.00144928], NP -> NP+DT PP [0.002
89855], NP -> NN NP|<NNP-NNP-CD-CD-CD> [0.00144928], NP -> NP+NNS NP|<PP-PP> [0.0
101449], NP -> JJ NP|<JJ-NN> [0.00144928], NP -> NP NP|<CC-PP-PP> [0.00144928], N
P -> NP SBAR+S+VP [0.00144928], NP -> NP+NNS PP [0.00434783], NP -> NP+NNS NP|<VP
-SBAR> [0.00144928], NP -> QP NNS [0.0101449], NP -> NP NNS [0.00144928], NP -> N
P+NNS VP [0.00289855], NP -> CD NP|<NNS-PP> [0.00144928], NP -> NP+NNS NP|<VP-ADV

P> [0.00434783], NP -> NP NP|<CC-NP+NNP> [0.0057971], NP -> NP+NNS NP|<PP-PP-NP> [0.00144928], NP -> CD NP|<CD-NNS> [0.00434783], NP -> JJ NP|<NNP-NNS> [0.00144928], NP -> JJ NP|<NN-NNS> [0.00724638], NP -> NNP NP|<CC-NNP> [0.00289855], NP -> JJS NP|<NN-NNS> [0.00144928], NP -> JJS NP|<NN-NN> [0.00144928], NP -> WDT NNS [0.00144928], NP -> NN NP|<NN-SYM> [0.00144928], NP -> NN NP|<NNP-NNP-CD-CD> [0.00144928], NP -> JJ NP|<NN-NN> [0.00144928], NP -> NN NP|<NNP-NNP-CD-CD-CD-CD> [0.00289855], NP -> NN NP|<NNP-NNP-CD-CD-CD-CD-CD> [0.00144928], NP -> JJS NN [0.0115942], NP -> JJS NP|<JJ-NN-NN> [0.00289855], NP -> JJ NP|<NN-JJ-NN-NN> [0.00144928], NP -> DT NP|<CD-NN-NN> [0.00144928], NP -> CC NP|<NNP-CC-NNP> [0.00144928], NP -> NN NP|<NN-NNS> [0.00434783], NP -> NN NP|<NNP-NNP> [0.00144928], NP -> DT NP|<ADJP-NN> [0.00144928], NP -> NP NP [0.00289855], NP -> CD NP|<NN-NNS> [0.00289855], NP -> CD NP|<RB-RB> [0.0057971], NP -> NNP NNS [0.00144928], NP -> NP+NNS NP|<PP-PP-PP> [0.00144928], NP -> QP NP|<NNS-QP> [0.00144928], NP -> DT NP|<JJS-NNS> [0.00144928], NP -> DT NP|<JJS-JJ-NN-NNS> [0.00144928], NP -> DT NP|<JJ-NNS> [0.00144928], NP -> DT NP|<NNP-NNP-NNS> [0.00289855], NP -> RB NP|<DT-NNP-NNP-NNS> [0.00144928], NP -> DT NP|<NNP-NNP-NNP-NNS> [0.00144928], NP -> DT NP|<NNP-NNP-NN> [0.00144928], NP -> DT NP|<NN-SYM-SYM-SYM-CD-CD> [0.00144928], NP -> DT NP|<NN-NN-SYM> [0.00144928], DT -> 'the' [0.626263], DT -> 'this' [0.0252525], DT -> 'a' [0.156566], DT -> 'The' [0.020202], DT -> 'those' [0.00505051], DT -> 'that' [0.020202], DT -> 'any' [0.00505051], DT -> 'these' [0.0555556], DT -> 'all' [0.0505051], DT -> 'Any' [0.00505051], DT -> 'each' [0.00505051], DT -> 'No' [0.00505051], DT -> 'All' [0.020202], NNS -> 'flights' [0.746411], NNS -> 'friends' [0.00478469], NNS -> 'ones' [0.00956938], NNS -> 'meals' [0.0191388], NNS -> 'details' [0.00478469], NNS -> 'numbers' [0.0143541], NNS -> 'fares' [0.0717703], NNS -> 'airlines' [0.0143541], NNS -> 'stops' [0.0191388], NNS -> 'stopovers' [0.00956938], NNS -> 'minutes' [0.0239234], NNS -> 'hours' [0.0191388], NNS -> 'restrictions' [0.00478469], NNS -> 'mornings' [0.00478469], NNS -> 'dollars' [0.0191388], NNS -> 'codes' [0.00956938], NNS -> 'tickets' [0.00478469], NP|<PP-PP-SBAR> -> PP NP|<PP-SBAR> [1.0], PP -> IN NP+NNP [0.349927], PP -> TO NP+NNP [0.232796], PP -> IN NP [0.288433], PP -> IN NP+PRP [0.00146413], PP -> TO NP [0.0849195], PP -> IN NP+DT [0.0161054], PP -> IN NP+NN [0.0131772], PP -> IN ADJP+JJ [0.00146413], PP -> IN PP|<CC-IN-NP> [0.00146413], PP -> PP PP|<CC-PP> [0.00146413], PP -> NP PP|<IN-S+VP+VBG> [0.00146413], PP -> IN NP+NNPS [0.00146413], PP -> IN NP+NNS [0.00439239], PP -> TO INTJ+UH [0.00146413], IN -> 'from' [0.452282], IN -> 'in' [0.10166], IN -> 'on' [0.143154], IN -> 'of' [0.0809129], IN -> 'at' [0.0207469], IN -> 'around' [0.0124481], IN -> 'for' [0.0103734], IN -> 'Of' [0.00207469], IN -> 'after' [0.033195], IN -> 'about' [0.0124481], IN -> 'by' [0.00622407], IN -> 'between' [0.026971], IN -> 'with' [0.0124481], IN -> 'through' [0.00207469], IN -> 'On' [0.00829876], IN -> 'as' [0.00207469], IN -> 'via' [0.00622407], IN -> 'before' [0.026971], IN -> 'than' [0.0145228], IN -> 'within' [0.00207469], IN -> 'From' [0.0103734], IN -> 'With' [0.00622407], IN -> 'After' [0.00207469], IN -> 'during' [0.00207469], IN -> 'into' [0.00207469], NP+NNP -> 'Baltimore' [0.0421286], NP+NNP -> 'Seattle' [0.0133038], NP+NNP -> 'Minneapolis' [0.0221729], NP+NNP -> 'Wednesday' [0.0243902], NP+NNP -> 'Washington' [0.0177384], NP+NNP -> 'Denver' [0.0332594], NP+NNP -> 'Miami' [0.0332594], NP+NNP -> 'Indianapolis' [0.0243902], NP+NNP -> 'Houston' [0.0354767], NP+NNP -> 'Atlanta' [0.0133038], NP+NNP -> 'Charlotte' [0.0199557], NP+NNP -> 'Tuesday' [0.00886918], NP+NNP -> 'Boston' [0.0133038], NP+NNP -> 'Orlando' [0.0310421], NP+NNP -> 'Milwaukee' [0.0421286], NP+NNP -> 'Tampa' [0.0310421], NP+NNP -> 'Nashville' [0.037694], NP+NNP -> 'Newark' [0.0620843], NP+NNP -> 'Phoenix' [0.0421286], NP+NNP -> 'Montreal' [0.0177384], NP+NNP -> 'Chicago' [0.0199557], NP+NNP -> 'Cincinnati' [0.0199557], NP+NNP -> 'Burbank' [0.0133038], NP+NNP -> 'American' [0.00443459], NP+NNP -> 'Tacoma' [0.0221729], NP+NNP -> 'July' [0.00665188], NP+NNP -> 'Alaska' [0.00221729], NP+NNP -> 'Delta' [0.00221729], NP+NNP -> 'Toronto' [0.0221729], NP+NNP -> 'Columbus' [0.0221729], NP+NNP -> 'Continental' [0.00221729], NP+NNP -> 'Sunday' [0.0177384], NP+NNP -> 'Monday' [0.0110865], NP+NNP -> 'Detroit' [0.0133038], NP+NNP -> 'Michigan' [0.00221729], NP+NNP -> 'Thursday' [0.00886918], NP+NNP -> 'Ontario' [0.0133038], NP+NNP -> 'Westchester' [0.00221729], NP+NNP -> 'Oakland' [0.0155211], NP+NNP -> 'Dallas' [0.0310421], NP+NNP -> 'Philadelphia' [0.0177384], NP+NNP -> 'Pittsburgh' [0.0266075], NP+NNP -> 'Saturday' [0.0155211], NP+NNP -> 'Memphis' [0.0310421], N

P+NNP -> 'Cleveland' [0.059867], NP+NNP -> 'Friday' [0.0199557], NP+NNP -> 'Tenne
ssee' [0.00221729], NP+NNP -> 'California' [0.00221729], NP+NNP -> 'Florida' [0.0
0221729], NP+NNP -> 'Arrival' [0.00221729], NP+NNP -> 'Dulles' [0.00221729], NP|<
PP-SBAR> -> PP SBAR [1.0], TO -> 'to' [1.0], SBAR -> WHNP+WDT S+VP [0.909091], SB
AR -> WHNP+WDT S [0.0454545], SBAR -> WHNP SQ [0.0454545], WHNP+WDT -> 'that' [0.
428571], WHNP+WDT -> 'Which' [0.571429], VBP -> 'stop' [0.0350877], VBP -> 'need'
[0.140351], VBP -> 'have' [0.0614035], VBP -> 'are' [0.236842], VBP -> 'leave'
[0.0964912], VBP -> 'do' [0.0438596], VBP -> 'make' [0.0175439], VBP -> 'am' [0.0
0877193], VBP -> 'Do' [0.0350877], VBP -> 'prefer' [0.00877193], VBP -> 'go' [0.0
789474], VBP -> "'m" [0.00877193], VBP -> 'serve' [0.0438596], VBP -> 'want' [0.0
263158], VBP -> 'mean' [0.00877193], VBP -> 'plan' [0.00877193], VBP -> 'depart'
[0.0614035], VBP -> 'fly' [0.0175439], VBP -> 'arrive' [0.0350877], VBP -> 'Are'
[0.00877193], VBP -> "'re" [0.00877193], VBP -> 'accept' [0.00877193], PUNC ->
'.' [0.73774], PUNC -> '?' [0.26226], SQ -> VBZ SQ|<NP-VP> [0.0933333], SQ -> MD
SQ|<NP-VP+VB> [0.0266667], SQ -> VBP NP [0.0666667], SQ -> VBP SQ|<NP+PRP-VP> [0.
04], SQ -> VBZ NP+PRP [0.0266667], SQ -> VBZ NP [0.293333], SQ -> VBP SQ|<NP-VP>
[0.0266667], SQ -> VBP SQ|<NP-VP+VB> [0.0266667], SQ -> VBZ SQ|<NP+NN-VP> [0.0133
333], SQ -> VBZ SQ|<NP-PP> [0.0133333], SQ -> VBZ SQ|<NP+NP+EX-NP> [0.0266667], S
Q -> MD SQ|<NP+PRP-VP> [0.0266667], SQ -> VBP SQ|<NP+NP+EX-NP-PP-PP> [0.0266667],
SQ -> X SQ|<VBZ-NP-ADJP+JJ> [0.0133333], SQ -> VBZ SQ|<NP-NP> [0.0266667], SQ ->
VBP NP+NP+EX [0.0133333], SQ -> VBP SQ|<NP+DT-NP> [0.0133333], SQ -> VBP SQ|<NP+N
P+EX-PP-PP-PP> [0.0266667], SQ -> VBP SQ|<NP+NP+EX-PP-PP> [0.173333], SQ -> VBP S
Q|<NP+PRP-VP+VB> [0.0133333], SQ -> INTJ+UH SQ|<MD-NP+PRP-VP> [0.0133333], VBZ ->
'Does' [0.12069], VBZ -> 'lives' [0.0344828], VBZ -> 'goes' [0.0172414], VBZ ->
'arrives' [0.0344828], VBZ -> 'is' [0.534483], VBZ -> 'Is' [0.12069], VBZ -> 'lea
ves' [0.0689655], VBZ -> 'serves' [0.0344828], VBZ -> 'departs' [0.0172414], VBZ
-> 'has' [0.0172414], SQ|<NP-VP> -> NP VP [1.0], NN -> 'flight' [0.266904], NN ->
'stop' [0.024911], NN -> 'friend' [0.00355872], NN -> 'return' [0.00711744], NN -
> 'class' [0.0498221], NN -> 'fare' [0.0676157], NN -> 'coach' [0.00711744], NN -
> 'number' [0.0177936], NN -> 'ground' [0.0177936], NN -> 'transportation' [0.024
911], NN -> 'time' [0.00711744], NN -> 'one' [0.00711744], NN -> 'aircraft' [0.00
711744], NN -> 'afternoon' [0.0284698], NN -> 'today' [0.00355872], NN -> 'evenin
g' [0.0391459], NN -> 'tomorrow' [0.0106762], NN -> 'cost' [0.00711744], NN -> 't
rip' [0.0604982], NN -> 'type' [0.00711744], NN -> 'reservation' [0.00711744], NN
-> 'discount' [0.00355872], NN -> 'stopover' [0.00355872], NN -> 'layover' [0.003
55872], NN -> 'leg' [0.00711744], NN -> 'morning' [0.0604982], NN -> 'day' [0.010
6762], NN -> 'airline' [0.00711744], NN -> 'Flight' [0.0106762], NN -> 'dinner'
[0.0213523], NN -> 'way' [0.0284698], NN -> 'lunch' [0.00711744], NN -> 'abbrevia
tion' [0.00355872], NN -> 'snack' [0.00355872], NN -> 'noon' [0.00711744], NN ->
'night' [0.00711744], NN -> 'nonstop' [0.00711744], NN -> 'Morning' [0.00355872],
NN -> 'price' [0.0391459], NN -> 'code' [0.00711744], NN -> 'seating' [0.0035587
2], NN -> 'capacity' [0.00355872], NN -> 'kind' [0.00355872], NN -> 'airfare' [0.
0177936], NN -> 'ticket' [0.00355872], NN -> 'Return' [0.00711744], NN -> 'busine
ss' [0.00711744], NN -> 'Business' [0.00355872], NN -> 'Ground' [0.0106762], NN -
> 'Weekday' [0.00355872], NN -> 'Coach' [0.0106762], NN -> 'meal' [0.00355872], N
N -> 'restriction' [0.00355872], NN -> 'flyer' [0.00355872], VP -> VB NP+NN [0.04
4586], VP -> VBP NP [0.10828], VP -> VBG PP [0.0955414], VP -> VBG NP [0.038216
6], VP -> VBP S+VP [0.0509554], VP -> VB S [0.0127389], VP -> VB S+VP [0.082802
5], VP -> VB VP|<NP+PRP-PP-ADVP> [0.00636943], VP -> VBZ PP [0.0127389], VP -> VB
VP|<NP+PRP-ADVP> [0.00636943], VP -> MD VP [0.121019], VP -> VB VP|<PP-NP+NN> [0.
00636943], VP -> VB VP|<PP-PP> [0.0509554], VP -> VB PP [0.044586], VP -> VB NP
[0.0955414], VP -> TO VP [0.00636943], VP -> VP VP|<CC-VP> [0.0191083], VP -> VB
VP|<PRT+RP-NP> [0.00636943], VP -> VBG NP+NNP [0.00636943], VP -> VBP PP [0.03184
71], VP -> VBN PP [0.0191083], VP -> VBP ADJP [0.00636943], VP -> VB NP+NNP [0.01
27389], VP -> VB VP|<PP-X+TO-PP> [0.00636943], VP -> VB VP|<ADVP-NP> [0.0063694
3], VP -> VBD NP [0.00636943], VP -> VB VP|<ADVP+RB-PP-ADJP> [0.00636943], VP ->
VB VP|<PP-PP-NP> [0.00636943], VP -> VB VP|<NP+PRP-NP> [0.0191083], VP -> VBP FRA
G [0.00636943], VP -> VB VP|<PP-PP-PP> [0.00636943], VP -> VBP VP|<NP+NNP-PP> [0.
00636943], VP -> VBP VP|<PP-PP> [0.00636943], VP -> VB NP+NNS [0.00636943], VP ->
VBG VP|<PP-PP> [0.0191083], VP -> VBG VP|<PP-PP-PP> [0.00636943], VP -> VBZ NP

[0.00636943], NP+NN -> 'dinner' [0.27451], NP+NN -> 'tomorrow' [0.117647], NP+NN -> 'noon' [0.117647], NP+NN -> 'today' [0.0196078], NP+NN -> 'lunch' [0.0588235], NP+NN -> 'Flight' [0.0784314], NP+NN -> 'flight' [0.0588235], NP+NN -> 'stay' [0.0196078], NP+NN -> 'Airline' [0.0392157], NP+NN -> 'aircraft' [0.0196078], NP+NN -> 'airplane' [0.0196078], NP+NN -> 'Price' [0.137255], NP+NN -> 'price' [0.0196078], NP+NN -> 'Departure' [0.0196078], S -> NP+PRP VP [0.590164], S -> NP+NN VP+VBN [0.0163934], S -> S S|<CC-S> [0.0327869], S -> NP VP [0.131148], S -> INTJ+UH VP [0.0655738], S -> INTJ+UH S|<NP+PRP-VP> [0.0163934], S -> NP+NNP VP+VBZ [0.0163934], S -> PP S|<NP+PRP-VP> [0.0327869], S -> NP+PRP S|<ADVP+RB-VP> [0.0163934], S -> ADVP+RB S|<NP+PRP-VP> [0.0327869], S -> ADVP+RB VP [0.0327869], S -> NP+NN VP [0.0163934], NP+PRP -> 'I' [0.323529], NP+PRP -> 'me' [0.602941], NP+PRP -> 'them' [0.00735294], NP+PRP -> 'you' [0.0441176], NP+PRP -> 'it' [0.0147059], NP+PRP -> 'they' [0.00735294], NP|<PP-VP-VP> -> PP NP|<VP-VP> [1.0], NP|<VP-VP> -> VP VP [1.0], VBG -> 'leaving' [0.264706], VBG -> 'making' [0.0294118], VBG -> 'living' [0.0294118], VBG -> 'arriving' [0.352941], VBG -> 'Leaving' [0.117647], VBG -> 'Arriving' [0.0294118], VBG -> 'departing' [0.0882353], VBG -> 'Traveling' [0.0294118], VBG -> 'Departing' [0.0588235], VP+VBN -> 'served' [1.0], CD -> 'two' [0.0142857], CD -> 'eleven' [0.0142857], CD -> 'five' [0.128571], CD -> 'seven' [0.0785714], CD -> 'ten' [0.0785714], CD -> 'fifty' [0.0428571], CD -> 'six' [0.0785714], CD -> 'oh' [0.0142857], CD -> 'eighteen' [0.00714286], CD -> 'twenty' [0.0142857], CD -> 'twelve' [0.0357143], CD -> 'fourteen' [0.00714286], CD -> 'thirty' [0.0785714], CD -> 'one' [0.185714], CD -> 'four' [0.0214286], CD -> 'nine' [0.05], CD -> 'zero' [0.00714286], CD -> 'eight' [0.0285714], CD -> 'seventeen' [0.00714286], CD -> 'hundred' [0.0571429], CD -> 'forty' [0.0142857], CD -> 'thousand' [0.00714286], CD -> 'three' [0.0142857], CD -> 'nineteen' [0.00714286], CD -> 'Zero' [0.00714286], MD -> 'would' [0.346154], MD -> 'should' [0.153846], MD -> "'d" [0.384615], MD -> 'Can' [0.0769231], MD -> 'could' [0.0384615], VP|<NP+PRP-PP-ADVP> -> NP+PRP VP|<PP-ADVP> [1.0], VP|<PP-ADVP> -> PP ADVP [1.0], ADVP -> ADVP+RB PP [0.285714], ADVP -> RB PP [0.142857], ADVP -> NP RB [0.571429], ADVP+RB -> 'here' [0.125], ADVP+RB -> 'home' [0.0625], ADVP+RB -> 'also' [0.0625], ADVP+RB -> 'actually' [0.0625], ADVP+RB -> 'Now' [0.125], ADVP+RB -> 'last' [0.0625], ADVP+RB -> 'first' [0.0625], ADVP+RB -> 'Only' [0.125], ADVP+RB -> 'Anytime' [0.0625], ADVP+RB -> 'daily' [0.25], NNP -> 'D' [0.0122549], NNP -> 'C' [0.0147059], NNP -> 'T' [0.00980392], NNP -> 'W' [0.0147059], NNP -> 'A' [0.0245098], NNP -> 'North' [0.00245098], NNP -> 'Carolina' [0.00245098], NNP -> 'Monday' [0.0122549], NNP -> 'United' [0.0147059], NNP -> 'Airlines' [0.0416667], NNP -> 'San' [0.0735294], NNP -> 'Jose' [0.0171569], NNP -> 'New' [0.0612745], NNP -> 'Jersey' [0.00980392], NNP -> 'Ontario' [0.00245098], NNP -> 'International' [0.00490196], NNP -> 'Saturday' [0.00245098], NNP -> 'Los' [0.0269608], NNP -> 'Angeles' [0.0269608], NNP -> 'P' [0.00245098], NNP -> 'American' [0.0245098], NNP -> 'July' [0.00490196], NNP -> 'Diego' [0.0196078], NNP -> 'Kansas' [0.0343137], NNP -> 'City' [0.0710784], NNP -> 'York' [0.0514706], NNP -> 'Las' [0.0588235], NNP -> 'Vegas' [0.0588235], NNP -> 'Saint' [0.0196078], NNP -> 'Petersburg' [0.00735294], NNP -> 'Long' [0.00735294], NNP -> 'Beach' [0.00735294], NNP -> 'June' [0.00245098], NNP -> 'Tuesday' [0.00245098], NNP -> 'Francisco' [0.0367647], NNP -> 'Salt' [0.0147059], NNP -> 'Lake' [0.0147059], NNP -> 'Delta' [0.00245098], NNP -> 'Louis' [0.00735294], NNP -> 'N' [0.00490196], NNP -> 'Thursday' [0.0147059], NNP -> 'U' [0.00980392], NNP -> 'S' [0.00735294], NNP -> 'Air' [0.00735294], NNP -> 'Sunday' [0.00490196], NNP -> 'Wednesday' [0.00980392], NNP -> 'Friday' [0.00735294], NNP -> 'Cincinnati' [0.00245098], NNP -> 'Tampa' [0.00245098], NNP -> 'Westchester' [0.00980392], NNP -> 'County' [0.00980392], NNP -> 'Paul' [0.00490196], NNP -> 'Newark' [0.00245098], NNP -> 'Cleveland' [0.00490196], NNP -> 'Ohio' [0.00490196], NNP -> 'O' [0.00245098], NNP -> 'J' [0.00980392], NNP -> 'F' [0.0147059], NNP -> 'K' [0.00980392], NNP -> 'Milwaukee' [0.00245098], NNP -> 'Orlando' [0.00245098], NNP -> 'La' [0.00735294], NNP -> 'Guardia' [0.00735294], NNP -> 'Southwest' [0.00980392], NNP -> 'Dulles' [0.00245098], NP+CD -> 'one' [1.0], S|<CC-S> -> CC S [1.0], CC -> 'and' [0.804878], CC -> 'or' [0.170732], CC -> 'either' [0.0243902], JJ -> 'other' [0.030303], JJ -> 'next' [0.0909091], JJ -> 'first' [0.151515], JJ -> 'intereste d' [0.0151515], JJ -> 'ninth' [0.0151515], JJ -> 'round' [0.227273], JJ -> 'eighth' [0.0151515], JJ -> 'sixteenth' [0.0151515], JJ -> 'second' [0.0151515], JJ ->

'nonstop' [0.151515], JJ -> 'direct' [0.0151515], JJ -> 'same' [0.0151515], JJ -> 'many' [0.0454545], JJ -> 'Last' [0.030303], JJ -> 'last' [0.0151515], JJ -> 'Next' [0.0151515], JJ -> 'Early' [0.0151515], JJ -> 'First' [0.030303], JJ -> 'Round' [0.030303], JJ -> 'Direct' [0.0151515], JJ -> 'expensive' [0.0151515], JJ -> 'Daily' [0.0151515], JJ -> 'frequent' [0.0151515], NP|<VP-SBAR> -> VP SBAR [1.0], VP|<NP+PRP-ADVP> -> NP+PRP ADVP [1.0], NP|<PP-PP-PP> -> PP NP|<PP-PP> [1.0], NP|<PP-PP> -> PP PP [1.0], NP|<NNP-NNP> -> NNP NNP [1.0], VP|<PP-NP+NN> -> PP NP+NN [1.0], QP -> RB CD [0.214286], QP -> IN QP|<JJS-CD> [0.142857], QP -> RBR QP|<IN-CD> [0.285714], QP -> CD QP|<CD-CD> [0.142857], QP -> CC JJR [0.0714286], QP -> CD QP|<CD-CD-CD> [0.0714286], QP -> RB QP|<JJR-IN-CD-CD-CD> [0.0714286], RB -> 'around' [0.0566038], RB -> 'a.m' [0.150943], RB -> 'p.m' [0.415094], RB -> 'much' [0.0943396], RB -> 'back' [0.0188679], RB -> 'as' [0.0188679], RB -> 'early' [0.0188679], RB -> 'apart' [0.0754717], RB -> "o'clock" [0.0943396], RB -> 'no' [0.0188679], RB -> 'only' [0.0188679], RB -> 'not' [0.0188679], NP|<PP-PP-NP> -> PP NP|<PP-NP> [1.0], NP|<PP-NP> -> PP NP [1.0], VP|<PP-PP> -> PP PP [1.0], NP|<NN-NN> -> NN NN [1.0], SBARQ -> WHNP SQ [0.0784314], SBARQ -> PP SBARQ|<WHNP+WHNP-SQ+VP> [0.00980392], SBARQ -> WHNP+WP SQ [0.196078], SBARQ -> WHNP SQ+VP [0.411765], SBARQ -> WHNP+WHNP SQ [0.186275], SBARQ -> WHNP+WDT SQ [0.0490196], SBARQ -> WHADVP+WRB SQ [0.00980392], SBARQ -> WHNP+WDT SQ+VP [0.0588235], WHNP -> WHNP ADJP+JJ [0.010989], WHNP -> WRB RB [0.0549451], WHNP -> WHNP+WDT PP [0.186813], WHNP -> WDT NN [0.032967], WHNP -> WDT NNS [0.67033], WHNP -> WHNP PP [0.032967], WHNP -> WHADJP NNS [0.010989], WRB -> 'How' [1.0], ADJP+JJ -> 'extra' [0.111111], ADJP+JJ -> 'possible' [0.111111], ADJP+JJ -> 'last' [0.333333], ADJP+JJ -> 'nonstop' [0.222222], ADJP+JJ -> 'available' [0.111111], ADJP+JJ -> 'Nonstop' [0.111111], SQ|<NP-VP+VB> -> NP VP+VB [1.0], NP|<JJ-NN-NN> -> JJ NP|<NN-NN> [1.0], VP+VB -> 'cost' [0.5], VP+VB -> 'stop' [0.166667], VP+VB -> 'leave' [0.166667], VP+VB -> 'use' [0.166667], S+VP|<PP-PP> -> PP PP [1.0], VP|<CC-VP> -> CC VP [1.0], VP|<PRT+RP-NP> -> PRT+RP NP [1.0], PRT+RP -> 'out' [1.0], NP|<NNP-NN> -> NNP NN [1.0], S+VP|<NP+PRP-NP> -> NP+PRP NP [1.0], PDT -> 'all' [1.0], NP|<DT-NNS> -> DT NNS [1.0], SBARQ|<WHNP+WHNP-SQ+VP> -> WHNP+WHNP SQ+VP [1.0], WHNP+WHNP -> WDT NNS [0.9], WHNP+WHNP -> WHADJP NNS [0.1], WDT -> 'which' [0.0120482], WDT -> 'What' [0.915663], WDT -> 'what' [0.0120482], WDT -> 'Which' [0.060241], SQ+VP -> VBP PP [0.14], SQ+VP -> VP SQ+VP|<CC-VP> [0.04], SQ+VP -> VBP VP [0.02], SQ+VP -> VBP NP [0.08], SQ+VP -> VBZ NP+NN [0.02], SQ+VP -> VBP NP+NN [0.08], SQ+VP -> VBP ADVP+RB [0.02], SQ+VP -> VBZ ADJP+JJ [0.06], SQ+VP -> VBP ADVP+RBS [0.02], SQ+VP -> VBZ ADVP+RBS [0.02], SQ+VP -> VBP SQ+VP|<PP-PP> [0.12], SQ+VP -> VBP SQ+VP|<PP-PP-PP> [0.12], SQ+VP -> VBP PP+IN [0.02], SQ+VP -> VBP ADJP+JJ [0.02], SQ+VP -> VBP SQ+VP|<NP+NNP-PP> [0.06], SQ+VP -> VBP SQ+VP|<NP+NNP-PP-PP> [0.04], SQ+VP -> VBP SQ+VP|<PP-PP-NP> [0.02], SQ+VP -> VBP NP+NNS [0.02], SQ+VP -> VBZ NP [0.02], SQ+VP -> VBP NP+NNP [0.02], SQ+VP -> VBZ ADJP+JJR [0.04], WHNP+WP -> 'What' [1.0], NP+DT -> 'these' [0.533333], NP+DT -> 'this' [0.0666667], NP+DT -> 'any' [0.0666667], NP+DT -> 'those' [0.266667], NP+DT -> 'Any' [0.0666667], SQ+VP|<CC-VP> -> CC VP [1.0], S+VP|<NP+PRP-PP> -> NP+PRP PP [1.0], NP|<NN-NN-NN> -> NN NP|<NN-NN> [1.0], FRAG -> X PP [0.0689655], FRAG -> NP INTJ+UH [0.0344828], FRAG -> X NP [0.0689655], FRAG -> NP FRAG|<NP-NP-NP> [0.0344828], FRAG -> NP+NNP FRAG|<PP-PP> [0.0689655], FRAG -> PP VP [0.0344828], FRAG -> ADVP+RB NP+NNP [0.0344828], FRAG -> NP+NNP PP [0.172414], FRAG -> NP FRAG|<NP-PP-PP> [0.0344828], FRAG -> PP PP [0.137931], FRAG -> PP FRAG|<PP-VP> [0.0344828], FRAG -> WHNP PP [0.0344828], FRAG -> WHNP FRAG|<PP-PP> [0.0689655], FRAG -> NP+NN FRAG|<PP-PP-NP> [0.0344828], FRAG -> X NP+NNP [0.0344828], FRAG -> NP NP [0.0689655], FRAG -> NP+NNP FRAG|<PP-ADVP+RB> [0.0344828], X -> WP IN [0.666667], X -> WRB IN [0.166667], X -> VBZ NP+DT [0.166667], WP -> 'What' [0.75], WP -> 'what' [0.25], SQ|<NP+PRP-VP> -> NP+PRP VP [1.0], NP|<CC-CD-RB> -> CC NP|<CD-RB> [1.0], NP|<CD-RB> -> CD RB [1.0], NP|<JJ-NN> -> JJ NN [1.0], VBN -> 'used' [0.333333], VBN -> 'served' [0.666667], ADJP -> JJ PP [0.2], ADJP -> RB ADJP|<RB-PP> [0.2], ADJP -> RBS JJ [0.2], ADJP -> JJR PP [0.4], NX -> NX+NN NX|<CC-NX> [0.5], NX -> NN NN [0.5], NX+NN -> 'aircraft' [1.0], NX|<CC-NX> -> CC NX [1.0], NP|<PP-PP-NP+NN> -> PP NP|<PP-NP+NN> [1.0], NP|<PP-NP+NN> -> PP NP+NN [1.0], NP|<PP-PP-VP> -> PP NP|<PP-VP> [1.0], NP|<PP-VP> -> PP VP [1.0], VP|<PP-X+TO-PP> -> PP VP|<X+TO-PP> [1.0], VP|<X+TO-PP> -> X+TO PP [1.0], X+TO -> 'to' [1.0], WHADVP+WRB -> 'Where' [1.0], INTJ+UH -> 'Please' [0.25], INTJ+UH -> 'Okay' [0.166

667], INTJ+UH -> 'please' [0.166667], INTJ+UH -> 'Thanks' [0.166667], INTJ+UH -> 'Hi' [0.0833333], INTJ+UH -> 'oh' [0.0833333], INTJ+UH -> 'oops' [0.0833333], NP|<NNP-NNP-CD-CD-NN> -> NNP NP|<NNP-CD-CD-NN> [1.0], NP|<NNP-CD-CD-NN> -> NNP NP|<CD-CD-NN> [1.0], NP|<CD-CD-NN> -> CD NP|<CD-NN> [1.0], NP|<CD-NN> -> CD NN [1.0], S|<NP+PRP-VP> -> NP+PRP VP [1.0], NP|<CD-CD-CD> -> CD NP|<CD-CD> [1.0], NP|<CD-CD> -> CD CD [1.0], FRAG+NP -> NP PP [0.0571429], FRAG+NP -> NP FRAG+NP|<PP-PP> [0.152381], FRAG+NP -> NP FRAG+NP|<PP-PP-NP+NN> [0.00952381], FRAG+NP -> NP FRAG+NP|<PP-PP-PP> [0.0285714], FRAG+NP -> NP+NNS FRAG+NP|<PP-PP> [0.2], FRAG+NP -> NP+NNS FRAG+NP|<PP-PP-NP> [0.00952381], FRAG+NP -> NP+NNS FRAG+NP|<PP-PP-PP> [0.0380952], FRAG+NP -> NP+NNS FRAG+NP|<PP-PP-PP-PP> [0.0190476], FRAG+NP -> JJS NN [0.0380952], FRAG+NP -> NP VP+VBG [0.00952381], FRAG+NP -> NNP NN [0.0190476], FRAG+NP -> NP+NNS PP [0.0666667], FRAG+NP -> NN NNS [0.00952381], FRAG+NP -> NP+NNS FRAG+NP|<PP-PP-PP-VP> [0.00952381], FRAG+NP -> NP+NNS FRAG+NP|<PP-PP-VP> [0.00952381], FRAG+NP -> NP+NN PP [0.0571429], FRAG+NP -> NP FRAG+NP|<CC-NP+NN> [0.00952381], FRAG+NP -> NP+NN FRAG+NP|<PP-PP> [0.0285714], FRAG+NP -> NP+NN FRAG+NP|<PP-PP-SBAR> [0.00952381], FRAG+NP -> JJ FRAG+NP|<NN-NNS> [0.00952381], FRAG+NP -> DT NNS [0.00952381], FRAG+NP -> NN FRAG+NP|<NN-NNP-NNP-CD-CD-CD-CD> [0.00952381], FRAG+NP -> NN FRAG+NP|<NN-NN> [0.00952381], FRAG+NP -> CD NNS [0.00952381], FRAG+NP -> JJ NNS [0.00952381], FRAG+NP -> NN FRAG+NP|<NN-NNS> [0.0190476], FRAG+NP -> NP+NNP FRAG+NP|<PP-PP> [0.00952381], FRAG+NP -> JJS FRAG+NP|<JJ-NN-NN> [0.0190476], FRAG+NP -> NN NN [0.0285714], FRAG+NP -> NP+NNS FRAG+NP|<PP-NP> [0.00952381], FRAG+NP -> NP+NNS FRAG+NP|<PP-PP-NP+NNS> [0.00952381], FRAG+NP -> NP+NNS FRAG+NP|<PP-PP-ADVP+RB-NP> [0.00952381], FRAG+NP -> NP FRAG+NP|<PP-PP-VP> [0.00952381], FRAG+NP -> NP+NNS FRAG+NP|<PP-PP-NP-ADVP+RB> [0.00952381], FRAG+NP -> NP+NNS FRAG+NP|<PP-PP-ADVP+RB> [0.00952381], FRAG+NP -> NP FRAG+NP|<PP-ADJP> [0.00952381], FRAG+NP -> NP+NNS ADJP [0.00952381], FRAG+NP -> JJS NNS [0.00952381], NP|<NP+NN-PP-PP> -> NP+NN NP|<PP-PP> [1.0], NP|<NP+NN-PP-PP-PP> -> NP+NN NP|<PP-PP-PP> [1.0], NP|<JJS-NN> -> JJS NN [1.0], JJS -> 'longest' [0.03125], JJS -> 'latest' [0.125], JJS -> 'earliest' [0.09375], JJS -> 'least' [0.0625], JJS -> 'Earliest' [0.03125], JJS -> 'Shortest' [0.09375], JJS -> 'Cheapest' [0.40625], JJS -> 'Lowest' [0.03125], JJS -> 'lowest' [0.03125], JJS -> 'cheapest' [0.09375], SQ|<NP+NN-VP> -> NP+NN VP [1.0], NP|<CC-NP> -> CC NP [1.0], SQ|<NP-PP> -> NP PP [1.0], SQ|<NP+NP+EX-NP> -> NP+NP+EX NP [1.0], NP+NP+EX -> 'there' [1.0], NP|<NP-PP-X+TO-PP> -> NP NP|<PP-X+TO-PP> [1.0], NP|<PP-X+TO-PP> -> PP NP|<X+TO-PP> [1.0], NP|<X+TO-PP> -> X+TO PP [1.0], NP|<PP-PP-PP-NP+NN> -> PP NP|<PP-PP-NP+NN> [1.0], POS -> "'s" [1.0], NP|<NN-CD-CD-CD> -> NN NP|<CD-CD-CD> [1.0], NP|<NP-PP-PP> -> NP NP|<PP-PP> [1.0], NP|<NN-NNS> -> NN NNS [1.0], VP|<ADVP-NP> -> ADVP NP [1.0], VP+VBZ -> 'has' [1.0], NP|<CD-CD-CD-NN> -> CD NP|<CD-CD-NN> [1.0], VBD -> 'meant' [1.0], NP|<CD-CD-RB> -> CD NP|<CD-RB> [1.0], NP|<JJ-NN-NNS> -> JJ NP|<NN-NNS> [1.0], VP|<ADVP+RB-PP-ADJP> -> ADVP+RB VP|<PP-ADJP> [1.0], VP|<PP-ADJP> -> PP ADJP [1.0], ADJP|<RB-PP> -> RB PP [1.0], VP|<PP-PP-NP> -> PP VP|<PP-NP> [1.0], VP|<PP-NP> -> PP NP [1.0], VP|<NP+PRP-NP> -> NP+PRP NP [1.0], NP|<CC-ADVP+RB-NP> -> CC NP|<ADVP+RB-NP> [1.0], NP|<ADVP+RB-NP> -> ADVP+RB NP [1.0], FRAG|<NP-NP-NP> -> NP FRAG|<NP-NP> [1.0], FRAG|<NP-NP> -> NP NP [1.0], FRAG+NP|<PP-PP> -> PP PP [1.0], FRAG+NP+NN -> 'Tomorrow' [0.5], FRAG+NP+NN -> 'Cost' [0.5], FRAG+NP|<PP-PP-NP+NN> -> PP FRAG+NP|<PP-NP+NN> [1.0], FRAG+NP|<PP-NP+NN> -> PP NP+NN [1.0], FRAG+NP|<PP-PP-PP> -> PP FRAG+NP|<PP-PP> [1.0], S|<ADVP+RB-VP> -> ADVP+RB VP [1.0], FRAG|<PP-PP> -> PP PP [1.0], ADVP+RBS -> 'latest' [1.0], NP|<JJS-NN-NN> -> JJS NP|<NN-NN> [1.0], S+VP|<NP+PRP-NP-INTJ+UH> -> NP+PRP S+VP|<NP-INTJ+UH> [1.0], S+VP|<NP-INTJ+UH> -> NP INTJ+UH [1.0], WHADJP -> WRB JJ [1.0], SQ|<NP+NP+EX-NP-PP-PP> -> NP+NP+EX SQ|<NP-PP-PP> [1.0], SQ|<NP-PP-PP> -> NP SQ|<PP-PP> [1.0], SQ|<PP-PP> -> PP PP [1.0], SQ+VP|<PP-PP> -> PP PP [1.0], SQ+VP|<PP-PP-PP> -> PP SQ+VP|<PP-PP> [1.0], X+SBARQ -> WHNP SQ+VP [1.0], PP+IN -> 'from' [1.0], NP|<NN-SYM-CD> -> NN NP|<SYM-CD> [1.0], NP|<SYM-CD> -> SYM CD [1.0], SYM -> 'D' [0.166667], SYM -> 'H' [0.166667], SYM -> 'A' [0.166667], SYM -> 'P' [0.166667], SYM -> 'slash' [0.166667], SYM -> 'Q' [0.166667], QP|<JJS-CD> -> JJS CD [1.0], VP|<PP-PP-PP> -> PP VP|<PP-PP> [1.0], NP|<SBAR-PP> -> SBAR PP [1.0], VP|<NP+NNP-PP> -> NP+NNP PP [1.0], SQ+VP|<NP+NNP-PP> -> NP+NNP PP [1.0], SQ|<VBZ-NP-ADJP+JJ> -> VBZ SQ|<NP-ADJP+JJ> [1.0], SQ|<NP-ADJP+JJ> -> NP ADJP+JJ [1.0], SQ|<NP-NP> -> NP NP [1.0], NP|<NNP-NN-CD-CD> -> NNP NP|<NN-CD-CD> [1.0], NP|<NN-CD-CD> -> NN NP|<CD-CD> [1.0], NP|<JJ-X+JJ-NNS> -> JJ NP|<X+JJ-NNS>

[1.0], NP|<X+JJ-NNS> -> X+JJ NNS [1.0], X+JJ -> 'nonstop' [1.0], NP+NNS -> 'stop
s' [0.0144928], NP+NNS -> 'flights' [0.231884], NP+NNS -> 'Flights' [0.594203], N
P+NNS -> 'costs' [0.0144928], NP+NNS -> 'meals' [0.0144928], NP+NNS -> 'Wednesday
s' [0.0144928], NP+NNS -> 'weekdays' [0.0434783], NP+NNS -> 'Airports' [0.028985
5], NP+NNS -> 'Fares' [0.0289855], NP+NNS -> 'fares' [0.0144928], SQ|<NP+DT-NP> -
> NP+DT NP [1.0], SQ|<NP+NP+EX-PP-PP-PP> -> NP+NP+EX SQ|<PP-PP-PP> [1.0], SQ|<PP-
PP-PP> -> PP SQ|<PP-PP> [1.0], SQ|<NP+NP+EX-PP-PP> -> NP+NP+EX SQ|<PP-PP> [1.0],
SQ+VP|<NP+NNP-PP-PP> -> NP+NNP SQ+VP|<PP-PP> [1.0], NP|<NNP-NNP-CD-CD-CD> -> NNP
NP|<NNP-CD-CD-CD> [1.0], NP|<NNP-CD-CD-CD> -> NNP NP|<CD-CD-CD> [1.0], FRAG+VP ->
VBG FRAG+VP|<NP+NNP-PP> [0.125], FRAG+VP -> VBG PP [0.125], FRAG+VP -> VBG FRAG+V
P|<NP-PP> [0.25], FRAG+VP -> VBG NP [0.125], FRAG+VP -> VBG FRAG+VP|<PP-PP> [0.37
5], FRAG+VP|<NP+NNP-PP> -> NP+NNP PP [1.0], FRAG+NP|<PP-PP-NP> -> PP FRAG+NP|<PP-
NP> [1.0], FRAG+NP|<PP-NP> -> PP NP [1.0], FRAG+NP|<PP-PP-PP-PP> -> PP FRAG+NP|<P
P-PP-PP> [1.0], VP+VBG -> 'returning' [1.0], PP|<CC-IN-NP> -> CC PP|<IN-NP> [1.
0], PP|<IN-NP> -> IN NP [1.0], NP|<CC-PP-PP> -> CC NP|<PP-PP> [1.0], SBAR+S+VP ->
TO VP+VB [1.0], PP|<CC-PP> -> CC PP [1.0], RBR -> 'less' [1.0], QP|<IN-CD> -> IN
CD [1.0], NP|<NNS-PP> -> NNS PP [1.0], NP|<VP-ADVP> -> VP ADVP [1.0], NP|<CC-NP+N
NP> -> CC NP+NNP [1.0], NP|<CD-NNS> -> CD NNS [1.0], FRAG|<NP-PP-PP> -> NP FRAG|<
PP-PP> [1.0], NP|<NNP-NNS> -> NNP NNS [1.0], FRAG+NP+NNP -> 'Tuesday' [1.0], SQ+V
P|<PP-PP-NP> -> PP SQ+VP|<PP-NP> [1.0], SQ+VP|<PP-NP> -> PP NP [1.0], NP|<CC-NNP>
-> CC NNP [1.0], FRAG+VP|<NP-PP> -> NP PP [1.0], FRAG|<PP-VP> -> PP VP [1.0], FRA
G+WHNP -> WHNP FRAG+WHNP|<PP-PP-PP> [0.393939], FRAG+WHNP -> NP FRAG+WHNP|<PP-PP-
PP> [0.030303], FRAG+WHNP -> WHNP FRAG+WHNP|<PP-PP> [0.393939], FRAG+WHNP -> WHNP
FRAG+WHNP|<PP-PP-PP-PP> [0.121212], FRAG+WHNP -> WHNP FRAG+WHNP|<PP-PP-PP-PP-PP>
[0.030303], FRAG+WHNP -> WHNP PP [0.030303], FRAG+WHNP|<PP-PP-PP> -> PP FRAG+WHNP
|<PP-PP> [1.0], FRAG+WHNP|<PP-PP> -> PP PP [1.0], FRAG+WHNP|<PP-PP-PP-PP> -> PP F
RAG+WHNP|<PP-PP-PP> [1.0], FRAG+WHNP|<PP-PP-PP-PP-PP> -> PP FRAG+WHNP|<PP-PP-PP-P
P> [1.0], NP|<NN-SYM> -> NN SYM [1.0], NP|<NNP-NNP-CD-CD> -> NNP NP|<NNP-CD-CD>
[1.0], NP|<NNP-CD-CD> -> NNP NP|<CD-CD> [1.0], FRAG+PP -> IN NP+NNP [0.2], FRAG+P
P -> IN NP [0.8], SQ|<NP+PRP-VP+VB> -> NP+PRP VP+VB [1.0], FRAG+NP|<PP-PP-PP-VP>
-> PP FRAG+NP|<PP-PP-VP> [1.0], FRAG+NP|<PP-PP-VP> -> PP FRAG+NP|<PP-VP> [1.0], F
RAG+NP|<PP-VP> -> PP VP [1.0], FRAG|<PP-PP-NP> -> PP FRAG|<PP-NP> [1.0], FRAG|<PP
-NP> -> PP NP [1.0], NP|<NNP-NNP-CD-CD-CD-CD> -> NNP NP|<NNP-CD-CD-CD-CD> [1.0],
NP|<NNP-CD-CD-CD-CD> -> NNP NP|<CD-CD-CD-CD> [1.0], NP|<CD-CD-CD-CD> -> CD NP|<CD
-CD-CD> [1.0], FRAG+NP|<CC-NP+NN> -> CC NP+NN [1.0], NP|<NNP-NNP-CD-CD-CD-CD-CD>
-> NNP NP|<NNP-CD-CD-CD-CD-CD> [1.0], NP|<NNP-CD-CD-CD-CD-CD> -> NNP NP|<CD-CD-CD
-CD-CD> [1.0], NP|<CD-CD-CD-CD-CD> -> CD NP|<CD-CD-CD-CD> [1.0], FRAG+NP|<PP-PP-S
BAR> -> PP FRAG+NP|<PP-SBAR> [1.0], FRAG+NP|<PP-SBAR> -> PP SBAR [1.0], PP|<IN-S+
VP+VBG> -> IN S+VP+VBG [1.0], S+VP+VBG -> 'arriving' [1.0], NP|<NN-JJ-NN-NN> -> N
N NP|<JJ-NN-NN> [1.0], SQ|<MD-NP+PRP-VP> -> MD SQ|<NP+PRP-VP> [1.0], NP|<CD-NN-NN
> -> CD NP|<NN-NN> [1.0], NP|<NNP-CC-NNP> -> NNP NP|<CC-NNP> [1.0], ADJP+JJR ->
'better' [1.0], FRAG+NP|<NN-NNS> -> NN NNS [1.0], FRAG+NP|<NN-NNP-NNP-CD-CD-CD-CD
> -> NN FRAG+NP|<NNP-NNP-CD-CD-CD-CD> [1.0], FRAG+NP|<NNP-NNP-CD-CD-CD-CD> -> NNP
FRAG+NP|<NNP-CD-CD-CD-CD> [1.0], FRAG+NP|<NNP-CD-CD-CD-CD> -> NNP FRAG+NP|<CD-CD-
CD-CD> [1.0], FRAG+NP|<CD-CD-CD-CD> -> CD FRAG+NP|<CD-CD-CD> [1.0], FRAG+NP|<CD-C
D-CD> -> CD FRAG+NP|<CD-CD> [1.0], FRAG+NP|<CD-CD> -> CD CD [1.0], FRAG+NP|<NN-NN
> -> NN NN [1.0], FRAG+ADJP+JJ -> 'Nonstop' [1.0], NP|<ADJP-NN> -> ADJP NN [1.0],
RBS -> 'least' [1.0], FRAG+NP|<JJ-NN-NN> -> JJ FRAG+NP|<NN-NN> [1.0], FRAG+VP|<PP
-PP> -> PP PP [1.0], NP+NNPS -> 'Wednesdays' [1.0], FRAG+ADJP+JJS -> 'Cheapest'
[1.0], NP|<RB-RB> -> RB RB [1.0], FRAG+NP|<PP-PP-NP+NNS> -> PP FRAG+NP|<PP-NP+NNS
> [1.0], FRAG+NP|<PP-NP+NNS> -> PP NP+NNS [1.0], FRAG+NP|<PP-PP-ADVP+RB-NP> -> PP
FRAG+NP|<PP-ADVP+RB-NP> [1.0], FRAG+NP|<PP-ADVP+RB-NP> -> PP FRAG+NP|<ADVP+RB-NP>
[1.0], FRAG+NP|<ADVP+RB-NP> -> ADVP+RB NP [1.0], FRAG+NP|<PP-PP-NP-ADVP+RB> -> PP
FRAG+NP|<PP-NP-ADVP+RB> [1.0], FRAG+NP|<PP-NP-ADVP+RB> -> PP FRAG+NP|<NP-ADVP+RB>
[1.0], FRAG+NP|<NP-ADVP+RB> -> NP ADVP+RB [1.0], FRAG|<PP-ADVP+RB> -> PP ADVP+RB
[1.0], FRAG+NP|<PP-PP-ADVP+RB> -> PP FRAG+NP|<PP-ADVP+RB> [1.0], FRAG+NP|<PP-ADVP
+RB> -> PP ADVP+RB [1.0], FRAG+NP|<PP-ADJP> -> PP ADJP [1.0], JJR -> 'less' [0.7
5], JJR -> 'more' [0.25], QP|<CD-CD> -> CD CD [1.0], NP|<NNS-QP> -> NNS QP [1.0],
QP|<CD-CD-CD> -> CD QP|<CD-CD> [1.0], QP|<JJR-IN-CD-CD-CD> -> JJR QP|<IN-CD-CD-CD
> [1.0], QP|<IN-CD-CD-CD> -> IN QP|<CD-CD-CD> [1.0], NP|<JJS-NNS> -> JJS NNS [1.

```
0], S+VP|<CC-VP> -> CC VP [1.0], NP|<JJS-JJ-NN-NNS> -> JJS NP|<JJ-NN-NNS> [1.0],
NP|<JJ-NNS> -> JJ NNS [1.0], X+S+VP -> VB X+S+VP|<NP+PRP-NP> [1.0], X+S+VP|<NP+PR
P-NP> -> NP+PRP NP [1.0], NP|<NNP-NNP-NNS> -> NNP NP|<NNP-NNS> [1.0], NP|<DT-NNP-
NNP-NNS> -> DT NP|<NNP-NNP-NNS> [1.0], S+VP|<RB-VP> -> RB VP [1.0], NP|<NNP-NNP-N
NP-NNS> -> NNP NP|<NNP-NNP-NNS> [1.0], NP|<NNP-NNP-NN> -> NNP NP|<NNP-NN> [1.0],
NP|<NN-SYM-SYM-SYM-CD-CD> -> NN NP|<SYM-SYM-SYM-CD-CD> [1.0], NP|<SYM-SYM-SYM-CD-
CD> -> SYM NP|<SYM-SYM-CD-CD> [1.0], NP|<SYM-SYM-CD-CD> -> SYM NP|<SYM-CD-CD> [1.
0], NP|<SYM-CD-CD> -> SYM NP|<CD-CD> [1.0], NP|<NN-NN-SYM> -> NN NP|<NN-SYM> [1.
0]]
```

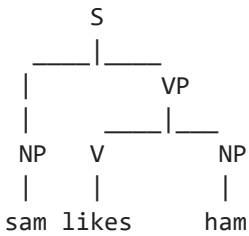
Part 5: Probabilistic CKY parsing

Finally, we are ready to implement probabilistic CKY parsing under PCFGs. Adapt the CKY parser from Part 3 to return the **most likely parse** and its **log probability** (base 2) given a PCFG. Note that to avoid underflows we want to work in the log space.

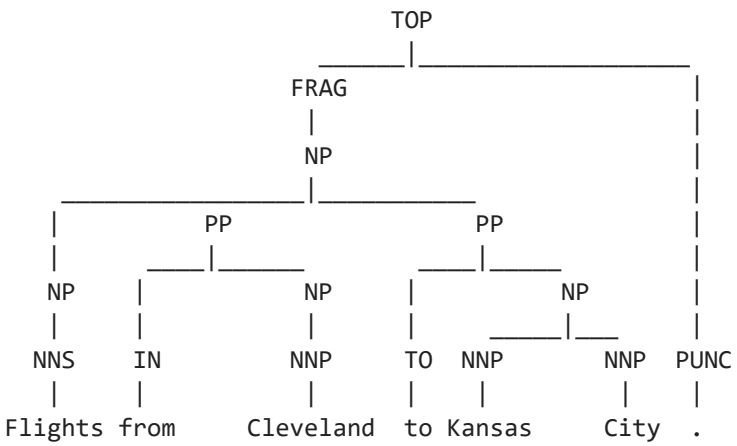
> **Hint:** `production.logprob()` will return the log probability of a PCFG production `production`.

As an aid in debugging, you may want to start by testing your implementation of probabilistic CKY on a much smaller grammar than the one you trained from the ATIS corpus. Here's a little grammar that you can play with.

> **Hint:** By "play with", we mean that you can change the grammar to try out the behavior of your parser on different test grammars, including ambiguous cases.




```
logprob: -2.25 | probability: 0.21
```




```
logprob:  -27 | probability: 7.42e-09
```

Evaluation of the grammar

There are a number of ways to evaluate parsing algorithms. In this project segment, you will use the "industry-standard" `evalb` implementation for computing constituent precision, recall, and F1 scores. We downloaded `evalb` during setup.

We read in the test data...

...and parse the test sentences using your probabilistic CKY implementation, writing the output trees to a file.

```
100%|██████████| 58/58 [00:02<00:00, 23.40it/s]
```

Now we can compare the predicted trees to the ground truth trees, using `evalb`. You should expect to achieve F1 of about 0.83.

```
data/outp.trees 345 brackets
data/test.trees 471 brackets
matching        339 brackets
precision       0.9826086956521739
recall  0.7197452229299363
F1      0.8308823529411764
```

Prompting Modern Large Language Models (LLMs)

Question: Modern large-scale language models (such as Claude, ChatGPT, Gemini, Llama) have various capabilities, that can be shown by prompting them correctly (i.e. giving them a correct input prompt). Try to see if you can prompt an LLM (of your choosing) to solve the task of **constituency parsing** on a few sample sentences (you can use the ones from this segement, or any other source of data). Write a short paragraph about your experience - what worked better, what worked worse. Note that your not expected to devise a fool-proof prompting method, but only to qualitatively experiment with prompting.

Using Gemini for constituency parsing, we provided prompts such as: "Parse the sentence: 'Sam likes ham' and return its constituency parse tree." Gemini effectively produced parse trees for straightforward sentences, demonstrating its capability to understand basic grammatical structures. However, for complex or ambiguous inputs, the model occasionally provided incomplete or incorrect parses. Simpler prompts with explicit instructions yielded better results. Collaborating with Tom and Yohan, we refined our approach by incrementally adjusting prompts, which noticeably improved Gemini's accuracy. This experiment underscored the need for clear and specific input to optimize LLM performance for parsing tasks.

Debrief

Question: We're interested in any thoughts you have about this project segment so that we can improve it for later years, and to inform later segments for this year. Please list any issues that arose or comments you have to improve the project segment. Useful things to comment on might include the following, but you are not restricted to these:

- Was the project segment clear or unclear? Which portions?
- Were the readings appropriate background for the project segment?
- Are there additions or changes you think would make the project segment better?

```
BEGIN QUESTION
name: open_response_debrief
manual: true
```

We worked together as a team (Tom, Yohan) and found this project segment overall enriching. However, on the technical side, the installation process for dependencies and ensuring compatibility with libraries like NLTK required extra effort.

Instructions for submission of the project segment

This project segment should be submitted to Gradescope, which will be made available some time before the due date.

Project segment notebooks are manually graded, not autograded using otter as labs are. (Otter is used within project segment notebooks to synchronize distribution and solution code however.) **We will not run your notebook before grading it.** Instead, we ask that you **submit the already freshly run notebook**. The best method is to "restart kernel and run all cells", allowing time for all cells to be run to completion. You should submit your code to Gradescope at the code submission assignment. **Make sure that you are also submitting your `data/grammar` file as part of your solution code as well.**

We also request that you **submit a PDF of the freshly run notebook**. The simplest method is to use "Export notebook to PDF", which will render the notebook to PDF via LaTeX. If that doesn't work, the method that seems to be most reliable is to export the notebook as HTML (if you are using Jupyter Notebook, you can do so using `File -> Print Preview`), open the HTML in a browser, and print it to a file. Then make sure to add the file to your git commit. Please name the file the same name as this notebook, but with a `.pdf` extension. (Conveniently, the methods just described will use that name by default.) You can then perform a git commit and push and submit the commit to Gradescope at the PDF submission assignment. **Make sure all of your solution pages are formatted correctly and that nothing got truncated when converting to PDF**.

End of project segment 3 {-}