Homework 4: Parsing

Name: Rashed Aldulijan

Class: 601.465/665− Natural Language Processing

Instructor: Prof. Jason Eisner

Due Date: Monday 20 October, 11 pm

Semester: Fall 2025

1(a): I tried parsing several sentences, and the parser performed as expected. It handled simple sentences accurately, producing well-structured phrase trees. It also managed more ambiguous examples, such as "*Papa ate the caviar with a spoon*" and "*Time flies like an arrow*," quite well

1(b): I tried some sentences in .sen files, I got:

1. (S (NP (DT The) (NN market)) (VP (VBZ is) (VP (VBG wondering) (SBAR (WHNP (WP what)) (S (NP (NNP General) (NNPS Motors)) (VP (VBZ has) (VP (VBN done))))))) (. .))
2. (S (PP (IN In) (NP (JJ recent) (NNS years))) (, ,) (NP (NN pay)) (VP (VBD surged) (SBAR (IN as) (S (NP (NN demand)) (VP (VBD rose) (SBAR (IN while) (S (NP (NNS workers)) (VP (VBD left) (PP (IN for) (NP (JJR easier) (NNS jobs)))))))))) (. .))
   We can see here that it did not do very well. *Demand rose* and *workers left for easier jobs* isn't parallal, that's *demand* is the noun phrase of the VP phrase following it, which isn't very accurate.
3. (S (NP (NN Papa)) (VP (VBD ate) (NP (DT the) (NN caviar)) (PP (IN with) (NP (DT the) (NN spoon))) (PP (IN with) (NP (NNP Papa))) (PP (IN with) (NP (NP (DT a) (NN spoon)) (PP (IN with) (NP (DT the) (NN caviar)))))))

It did well in all of the sentences that I tried except for 2.

1(c): I tried to parse *While the man hunted the deer ran away*. And it got a very wrong parsing: (S (SBAR (IN While) (NP (NP (DT the) (NN man)) (VP (VBD hunted) (NP (DT the) (NNS deer))))) (VP (VBD ran) (ADVP (RB away))) (. .))

2(a): I tested it with several sentences that we learned in class on https://demos.explosion.ai/displacy?text=The%20quick%20brown%20fox%20jumps%20over%20the%20lazy%20dog.&model=en_core_web_sm&cpu=1&cph=1
I found that this kind of parse try to find the first verb, and assign the words that depends on it before it and after it to this verb.

2(b): I found this Arabic parser, *https://corenlp.run/*,. Honstely, I was surprised from how well it did. In Arabic, the word structure is different from English.

Arabic Structure: V NP NP

English Structure: NP V NP

For example: Rashed ate the apple, in Arabic, the structure looks like, ate Rashed the apple, I tried a lot of sentences like *Rashed ate an apple with a spoon*, and *the horse that was raced past the barn fell.,* it really did very well, it correctly assigned fell to the horse, and with a spoon to the apple.

3:

Correctness:

In terms of printing the tree, when we run Earley algorithm, we need to store a back pointer for scan and attach to make use of it later when we find a tree.

When we need to know what the best derivation is, and we need to perform the following operations:

Predict: we initialize the weight of an item to 0.

Scan: we do not do anything.

Attach: because attach is the operation that would produce a new tree, we need to update the weight to have the current weight and the rule's own weight.

This will make the items or the list of "items" store the weight of the parse as well. As a result, the chart will have probabilities. We can now say choose the best parse in the last column of the chart.

Efficiency:

The $O(n^2)$ space bound follows because there are at most n + 1 chart columns, and each column stores a finite number of items proportional to the number of grammar symbols and positions in the sentence (i.e., O(n) items per column).

The $O(n^3)$ time bound arises because each item can generate a bounded number of predictions, scans, and attachments, and each of these steps considers at most O(n) positions—resulting in the classical cubic complexity.

4:

- E.1 Batch duplicate check:
    - Just added a list of sets to remember the set of items that already have been predicted at that position.
- E.2 Vocabulary specialization:
    - Before parsing, filter the grammar to temporarily remove any rules that contain terminal symbols not present in the current sentence
- E.3 Pruning
    - added a lightweight pruning mechanism that skips items whose cumulative weights are worse than the best one seen so far
- E.5 Indexing customers
    - I also attempted to implement customer indexing, where each chart column keeps an index mapping from the *next expected symbol* to the list of items waiting for it. However, my implementation didn't yield the expected results, so I removed this speedup.