Homework 3

CS 485, UMass Amherst, Fall 2023

Deliverables

Create your writeup as a PDF with any tool you like, and submit to Gradescope. Do not include large amounts of code in your writeup. (Small snippets may be appropriate if they help you answer a question in a concise and useful way.) Submit your code to a separate Gradescope assignment entry.

1 CKY Implementation

In this section, you will implement the CKY algorithm for an unweighted CFG. Start with the starter code *cky.py*.

1.1 CKY acceptance

Implement the acceptance version of CKY as `cky_acceptance()`, which returns `True` if there is an "S" covering the entire sentence (in other words, assume S is the only start symbol). Use the "GRAMMAR1" default grammar. Does it return `True` or `False` for the following sentences? Please `pprint()` the chart cells for each case as well.

- the the
- the table attacked a dog
- the cat

Hint: A simple way to implement the chart cells is by maintaining a list of nonterminals at the span. This list represents all possible nonterminals over that span.
Hint: `pprint()`ing the CKY chart cells may be useful for debugging.
Hint: Python dictionaries allow tuples as keys. For example, `d={}; d[(3,4)] = []`. A slight shortcut is that `d[3,4]` means the same thing as `d[(3,4)]`.

1.2 CKY parsing

Implement the parsing version of CKY, which returns one of the legal parses for the sentence (and returns None if there are none). If there are multiple real parses, we don't care which one you print. Implement this as `cky_parse()`. You probably want to start by copying your `cky_acceptance()` answer and modifying it. Have it return the parse in the following format, using nested lists to represent the tree (this is a simple Python variant of the Lisp-style S-expression that's usually used for this.)

```
['S',
        [['NP', [['Det', 'the'], ['Noun', 'cat']]],
         ['VP', [['Verb', 'attacked'],
                 ['NP', [['Det', 'the'], ['Noun', 'food']]]]]]]
```

Print out the parses for the following sentences.

- the cat saw a dog
- the cat saw a dog in a table
- the cat with a table attacked the food

Hint: In the chart cells, you will now have to store backpointers as well. One way to do it is to store a list of tuples, each of which is "(nonterminal, splitpoint, leftchild nonterm, rightchild nonterm)". For example, if the state "('NP', 3, 'Det', 'Noun')" is in the cell for span (2,4), that means this is a chart state of symbol NP, which came from a "Det" at position (2,3) and a Noun at position (3,4). *Hint*: It may be useful to use a recursive function for the backtrace.

1.3 Grammar improvement

Please revise the grammar as follows.

- Add four words to the lexicon: two verbs "attack" and "attacks", and the nouns "cats" and "dogs".
- Revise the rules to enforce subject-verb agreement on number.

The new grammar should accept and reject the following sentences. Please run your parser on these sentences and report the parse trees for the accepted ones. Also, describe how you changed the grammar, and why.

- ACCEPT: "the cat attacks the dog"
- REJECT: "the cat attack the dog"
- ACCEPT: "the cats attack the dog"
- REJECT: "the cat with the food on a dog attack the dog"

Hint: you will need to introduce new nonterminal symbols, and modify the currently existing ones.

1.4 Extra credit: Grammar engineering

Revise the grammar to incorporate coordination or other syntactic phenomena in English. Show examples of what it can analyze, and describe what you did.

2 Dependency parser errors

Run a dependency parser that uses the Universal Dependencies representation and try it out on some examples you make up. Feel free to use a web demo like from CoreNLP (https://corenlp.run/) or from SpaCy (https://demos.explosion.ai/displacy-ent) for this.

Show a sentence where the parser makes an error. Show the parse itself (for example, from a web visualization). Describe the error (including false positive edge(s) involved with the problem), and how to fix the error if you were to manually fix it.

Do this and draw a correct parse for the sentence.

Notes:

- We recommend trying shorter or intermediate length sentences, whose parses are easier to understand. But at the same time, longer sentences are more likely to have errors. You could start with something basic, then iteratively add more and more and see what happens.
- You'll want to read the Universal Dependencies documentation for this, at https://universaldependencies.org/
- As I showed in class, specifically for English relations the page is: https://universaldependencies.org/en/dep/index.html

3 Social Media Error Analysis for NER

Here, you'll use an off-the-shelf software package for NER. Feel free to use a web demo like from CoreNLP (https://corenlp.run/) or from SpaCy (https://demos.explosion.ai/displacy-ent). You can also download and run an NER package yourself; we recommend Stanza or SpaCy, which are both open-source Python libraries.

The NLP package you're using was probably not developed for social media. Let's test how well it does. This general problem is called *domain shift*, when something about the training data is systematically different than the data used at runtime; here, there's a shift in many things, including genre.

Run the system's named entity recognizer on twenty sentences from your HW2 annotation dataset. How well does it do at recognizing names? Read through the system's output and thinking about what it's getting right and wrong, before summarizing your findings. This is called *error analysis*. In fancier error analysis you can manually annotate the error types and do a statistical report; but here it's fine to be qualitative.

3.1 Error types and examples

Describe at least two types of errors the NER system tends to make on the text. For example, maybe there are certain types of entities it misses. For each error type, give an example. Make sure to describe both false positive and false negative errors.

3.2 Feature proposal

Propose a feature that might help fix some of the errors you observed, and explain why it might help. (This is a research hypothesis; if this were your project, you could then try actually doing it!)