Swarthmore Honors Project - Computational Linguistics
Spring 2019
Examiner: Jane Chandlee (jchandlee@haverford.edu)

Project summary: Implement the CKY sentence parsing algorithm in python, and then create a context-free grammar for Spanish using the parsed Spanish corpus provided by the Natural Language Toolkit (NLTK) software package. Use this grammar and the algorithm to try and recreate the parses in a Catalan corpus (also provided by NLTK). Assess the results and identify one or more needed changes to tailor the grammar from Spanish to Catalan.

Instructions

Step 1: Implement the CKY parsing algorithm. Read Chapter 11 (through section 11.2) of Speech and Language Processing (3rd edition) by Daniel Jurafsky and James H. Martin. Note: the 3rd edition is not published yet but a draft version is freely available here. Your first task is to implement the CKY algorithm in python, including the conversion to Chomsky Normal Form. You can test your implementation using the toy grammar in the text, but eventually you'll use a real grammar, which is Step 2.

But first, some tips on working with the NLTK corpora. You may or may not have worked with NLTK before. If you haven't, briefly familiarize yourself with it here and install it. The list of provided corpora can be found here. You'll be working specifically with two corpora: CESS-ESP (#91, a corpus of parsed Spanish sentences) and CESS-CAT (#75, a corpus of parsed Catalan sentences).

To get these corpora, once you have NLTK installed, open a python interpreter and do the following.

```
>>> import nltk
>>> nltk.download()
```

This will bring up a GUI from which you can select the corpora you want to download. Both are located under the 'Corpora' tab.

Once the corpora are downloaded do the following to access them:

```
>>> from nltk.corpus import cess_esp
>>> from nltk.corpus import cess_cat
```

The following gives you a list of the parsed sentences in bracket notation. (The second line will just show you what one looks like.)

```
>>> spanish_parsed = cess_esp.parsed_sents()
```

```
>>> print spanish_parsed[0]
```

To see an actual tree do:

```
>>> spanish_parsed[0].draw()
```

To get the unparsed sentences do:

```
>>> spanish_unparsed = cess_esp.sents()
```

Lastly, to get the sentences unparsed but with POS (part-of-speech) tagging do:

```
>>> spanish_POS = cess_esp.tagged_sents()
```

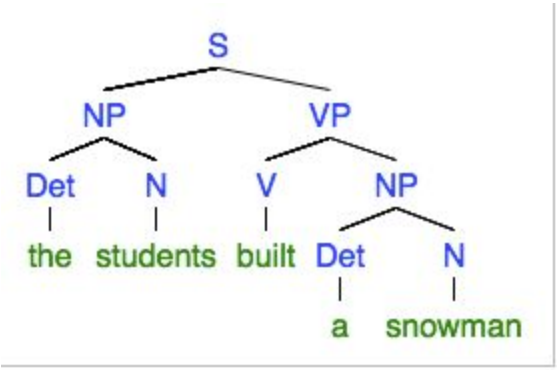
NOTE: The tagset for these corpora is based on the Eagles annotation scheme. It's not very intuitive, but ultimately you don't have to understand the tags themselves to do this project.

Step 2: Induce a CFG from parsed corpus. Carefully study the format of the parsed sentences in the ESP corpus and program a function that induces the rules of the context-free grammar. Here's a simple example:

Sentence: The students built a snowman.

Bracketed: Tree(S, [Tree(NP, [Tree(Det, [the])], [Tree(N, [students])])], [Tree(VP, [Tree(V, [built])], [Tree(NP, [Tree(Det, [a])], [Tree(N, [snowman])])])])

Tree:



Grammar rules:

S → NP VP
NP → Det N

VP → V NP
Det → the | a
N → students | snowman
V → built

Build a complete grammar for Spanish by identifying the rules for all sentences in the corpus (the grammar will be large!). Note you do not need to include the rules that list the actual lexical items (like the last three rules in the example above), because we are going to replace them in the next step.

Step 3. Add lexical items to grammar. Of course this grammar will not work on the Catalan data if it uses only lexical items from the Spanish corpus, so in this step you will create rules using the Catalan corpus that map each POS tag to its possible lexical items (i.e., rules of the form N → students | snowman). You can do this using the tagged but unparsed Catalan sentences (see above).

Step 4. Recreate parses. Now that you have your CKY algorithm and your grammar, run it on the unparsed/untagged Catalan sentences.

Step 5. Assess the results. This last step is open-ended. Minimally you want to automate a way to gauge the success of the grammar on the Catalan data (e.g., some function that provides an accuracy percentage when you compare the trees you got in Step 4 with the original trees in the Catalan corpus). From there, if you have time, come up with an automated way to do error analysis, or to discover for those trees that are not correct, what is wrong with them or what rules might need to be added to the grammar. You do NOT have to reach 100% accuracy by the end; the goal is to have a way to understand the degree to which a particular grammar fits a particular dataset.

Step 6: Write up your project. Please write up a brief report in which you describe what you did for the project, document any major hurdles and how you solved them, and present your final results.