

**LING 520: Computational Analysis of English**  
**Assignment: 4**  
**Due Date: March 31, 2008 by 11:00 p.m.**

---

## Information

### *Running the parser*

For this assignment, you will need to write a context free grammar to correctly parse a set of grammatical sentences and reject some ungrammatical ones.

To access the files, enter the following commands from your home directory on the linda server:

```
cp ~/520/assignments/a4.py ./
cp ~/520/assignments/gram.cfg ./
cp ~/520/assignments/test-sentences.txt ./
```

These files are write-protected. The only file you will need to make writable is `gram.cfg` by typing the following command:

```
chmod u+w gram.cfg
```

The file `a4.py` is simple parser interface that reads a grammar file specified as its command line argument, and parses input sentences provided through standard input. Therefore, you can run the program by typing this:

```
./a4.py gram.cfg
```

After issuing this command you will see the prompt `parser>` which looks for a line of input (or EOF). After a line of input is entered, the parser will attempt to parse it as a sentence and write the result on standard output. Once EOF is reached, the parser will calculate precision and recall of your grammar with respect to the input sentences and report those on standard output, as well.

In order for `a4.py` to calculate precision and recall correctly, ungrammatical sentences should be preceded by a `*`, and grammatical sentences with nothing, as in the following:

```
PARSER> the man saw the woman
Sentence: the man saw the woman
Parse 1:
(S (NP (Det the) (N man)) (VP (TV saw) (NP (Det the) (N woman))))
True positive.
PARSER> *man the woman saw
Sentence: *man the woman saw
True negative.
```

You are encouraged to review the code in `a4.py` to learn more about how it works.

## ***The grammar file***

The grammar file `gram.cfg` is a simple text file in which each line represents one CFG rule. Lines which begin with `#` are comments, and thus ignored.

The rules may include disjunctions (represented with `|`) on their right-hand-side as a shorthand. Therefore, the following line is equivalent to the following two lines:

```
Det -> 'the' | 'a'
```

```
Det -> 'the'
```

```
Det -> 'a'
```

Terminal nodes should be enclosed between single or double quotes. Non-terminal nodes must not include any quotes.

## ***Test sentences***

The file `test_sentences.txt` is a simple text file, in which each line represents a sentence to be parsed. Note that anything that follows a `#` is considered a comment and thus ignored. The ungrammatical sentences in this file are preceded by a `*`.

## **Directions**

You are required to review the sentences in `test_sentences.txt` and add CFG rules to the file `gram.cfg` to correctly parse those sentences. You will want to write rules that are as general as possible. In other words, your grammar should include the fewest number of rules possible and still provide the best parses.

**Note:** It is not possible to get 100% precision and recall on the test sentences with this formalism. But you should try to get the highest values with the most general rules.

**Hint:** A useful point to keep in mind is what is known in theoretical linguistics as the X-bar theory. The basic idea is that in recursive nodes, recursion happens at a level between lexical and maximal categories, and that modification happens at that medium level. For example, in order to account for nominal modification with adjective phrases, you may write rules like this:

```
NP -> Det Nbar    # We write Nbar as opposed to N' because we cannot
                  # use ' in non-lexical rules.
NP -> Nbar
Nbar -> AdjP Nbar
Nbar -> N
```

This way you account for all these phrases (assuming you have the right lexical items as well):

*car, the car, the black car, big black cars, those big cars, those identical big black cars, etc.*

You must examine the output for each sentence to make sure the parses you get are appropriate parses. You can also run the parser in batch mode by redirecting input and output as follows:

```
./a4.py gram.cfg < test_sentences.txt > parser_results.txt
```

This command makes a4.py read the input sentences from test\_sentences.txt and write the results in a file named parser\_results.txt.

## What to submit

You will need to submit both your grammar and results files. Before submitting, make sure you have an up-to-date version of **gram.cfg** and **parser\_results.txt** in your home directory. Then issue the following command from your home directory:

```
submit -a4 NETID
```

where NETID is your own net ID.