Homework #6: CMPT-413

Distributed on Mon, Mar 14; Due on Mon, Mar 21

Anoop Sarkar – anoop@cs.sfu.ca

(1) **Context-Free Grammar for ATIS** (Submit files: `testcky.pl`, `atis.cfg`)

Chapter 9 of Jurafsky & Martin contains a guide to the writing of a context-free grammar (CFG) for a fragment of English corresponding to the Air Traffic Information System (ATIS) corpus. Use this guide to write a context-free grammar for the sentences given in the file `atis.test` (available from the location specified on the course web page).

Write the CFG as a text file called `atis.cfg` in the following text file format. For example, a CFG:

$$
\begin{aligned}
S &\rightarrow AX \mid YB \\
X &\rightarrow AB \mid BA \\
Y &\rightarrow BA \\
A &\rightarrow a \\
B &\rightarrow a
\end{aligned}
$$

should be written as a text file (let's call it `ex1.cfg`) in the following format:

```
S A X
S Y B
X A B
X B A
Y B A
A a
B a
```

The non-terminal on the left-hand side of the first rule is assumed to be the start symbol for the grammar.

Get the Perl package `cky.pm` from the location specified on the course web page. It contains an implementation of the CKY parsing algorithm. The two functions from `cky.pm` that you will use are:

`readcfg` Takes one parameter which is the filename of the CFG in the text format explained above. This function returns an array containing two elements: a reference to the CFG data structure and a string containing the start symbol. Here is how to use this function:

```
my ($grammar, $start) = cky::readcfg("ex1.cfg");
```

For linguistic grammars, it is often inconvenient to write down a grammar in Chomsky Normal Form (CNF) which is required for CKY parsing. In order to deal with this issue, the function `readcfg` will accept CFGs that are not strictly in CNF. You can write a CFG in the following format:

> All rules are in the form $A \rightarrow a$ where A is a non-terminal and a is a terminal symbol or in the form $A \rightarrow \alpha$ where α is a sequence of one or more non-terminal symbols, e.g. $A \rightarrow B$ or $A \rightarrow ABCD$

The function `readcfg` automatically converts rules with more than two non-terminals in the right-hand side into CNF by inserting new non-terminals into the grammar. The CKY parser has been modified to deal with unary rules.

`parse` Takes three parameters, the grammar and the start symbol that are returned by the `readcfg` function above, and the third parameter contains a reference to an array containing the input string. The function returns a list of parses where each element is a parse tree for the input in the usual bracketed string format. For example:

```
@input = ('a', 'a', 'a');
my @parses = cky::parse($grammar, $start, \@input);
```

Create a new Perl program called `testcky.pl` which uses the `cky.pm` package to work with CFGs. Submit the file `testcky.pl` and make sure it has the appropriate command line arguments so that it can be tested as follows:

```
perl testcky.pl atis.cfg atis.test
```

The above command should use the CFG in the file `atis.cfg` that you have created in the text format explained above and print out all the parse trees for each sentence in `atis.test`. Do not worry about the extra non-terminals in the output parse trees that were inserted to create a CNF grammar.

Hint: To help you with some of the more difficult sentences in `atis.test`, I have placed a file that contains a single parse tree per sentence for a large number of ATIS sentences in the file `atis3.treebank`. Please do not make a copy of this file (it is under copyright).

You can use the Tcl/Tk script `viewtree` to view these trees in an X11 terminal, or you can use the perl script `indentrees.pl` to view the trees as indented text on any terminal.

```
wish viewtree < atis3.treebank
perl ../hw5/indentrees.pl < atis3.treebank | more
```