BM1 Advanced Natural Language Processing
University of Potsdam, Winter 2016/17

Assignment 3: CKY Parsing

Tatjana Scheffler, Ph.D.

Due: Friday, December 2, 10:00 a.m.

In this assignment, you will implement the CKY algorithm for English and apply it to the word recognition and parsing problem. You can use the NLTK modules for representing context-free grammars and parse trees, but you should implement the parser from scratch.

Ingredients. We provide the grammar and the test sentences.

The grammar stems from the Airline Travel Information System (ATIS), a project working on spoken dialog systems for air travel. The ATIS CFG is available in the NLTK data package, together with 98 test sentences. You can initialize the resources this way:

```
# load the grammar
grammar = nltk.data.load("grammars/large_grammars/atis.cfg")
# load the raw sentences
s = nltk.data.load("grammars/large_grammars/atis_sentences.txt", "raw")
# extract the test sentences
t = nltk.parse.util.extract_test_sentences(s)
```

NLTK already implements a number of parsing algorithms (see `nltk.parse` for the list). You can try one to see if you loaded the grammar correctly:

```
# initialize the parser
parser = nltk.parse.BottomUpChartParser(grammar)
# parse all test sentences
for sentence in t:
    parser.chart_parse(sentence[0])
```

However, the NLTK version of the ATIS grammar is not in Chomsky normal form (CNF), which you will need for your CKY parser. Feel free to implement a conversion module for extra credit, but for your convenience, we have already converted the ATIS CFG into CNF; you can download it from Moodle. You can then read the grammar from the file using `nltk.data.load()` and utilize the nice features of the `nltk.grammar` module on the resulting object.[1]

[1]Note that `chart_parse()` throws an error when it encounters an unknown word, which is undesirable behavior for any parser. If you want to test the grammars with the pre-implemented parser (for example, to check whether the CNF version is in fact weakly equivalent), you may need to catch this error.

Problem 1: Word recognition. Implement the CKY algorithm and use it as a recognizer. That is, given an input sentence, the procedure should decide whether the sentence is in the language of the CFG or not. Test the recognizer on the ATIS test sentences, but also by feeding it other sentences to see whether it properly rejects ungrammatical sentences as well.

Problem 2: Parsing. Now extend your CKY recognizer into a parser by adding backpointers. Also implement a function that extracts the set of all parse trees from the backpointers in the chart. Feel free to use the NLTK module `nltk.tree` for this purpose; notice that only `ImmutableTree`s can be used as elements of Python sets, whereas raw `Tree`s cannot.

Submission. Please submit your code, outputs, and a `README` file containing instructions for running your recognizer and parser. The outputs should consist of at least: (1) the list of ATIS test sentences with tab-separated numbers of parse trees, and (2) pictures of the parse trees for an ATIS test sentence of your choice with a number of parses p such that $1 < p < 5$. You can visualize an NLTK tree using its `draw` method.

Extra credit. If you still have time left, you can attempt the following project for extra credit. Perhaps it has occurred to you that it is quite wasteful to compute all parse trees just to find out how many parse trees there are. Figure out how to compute the number of parse trees for an entry $A \in \mathrm{Ch}(i, k)$ from your chart with backpointers, without actually computing these parse trees. Verify that you get the correct results, and compare the efficiency of your new procedure to your earlier solution.

Submit your solutions and code via email to `johannsmeier@uni-potsdam.de`