Computing the Probability of a Tree (stage 1)

Timon Kanters, Luisa Zintgraf, Philipp Beau

November 14, 2013

1 Task

In this assignment the focus was to understand and to implement how to calculate the log probability of a tree representing a piece of text given a grammar and a lexicon. In order to do this, we extended the given framework and implemented the method double getLogScore(Tree<String> tree) in BaselineCKYParser.

2 Implementation

In our solution method we went through the tree recursively summing the log scores of each subtree. It was important to understand to use the lexicon rules instead of the grammar rules depending on wether the current node is a preterminal or not. When the node is preterminal, we used the lexicon to retrieve the score for this node and the corresponding word. Otherwise we used the grammar to find the rule corresponding to the node and its direct children. To prevent the tree from being annotated multiple times during our recursion process, we moved the annotating operation outside of the getLogScore method. This means it is necessary to change line 104 in the class PCFGParserTester from

```
double logProb = parser.getLogScore(testTree);
```
to
```
double logProb = parser.getLogScore(annotator.annotateTree(testTree));
```

3 Results

The results of our final implementation can be seen in Table 1. Rules having a probability of 0, meaning they are not contained in the grammar, are represented with -Infinity.

0 logscore:	-50.99367882096118
1 logscore	-Infinity
2 logscore	-39.85517820793024
3 logscore	-41.67112250048548
4 logscore	-92.51659159264787
5 logscore	-Infinity
6 logscore	-101.62447403037064
7 logscore	-68.99276774663788
8 logscore	-55.70837907010919
9 logscore	-57.39394243941493
Total log prob	-Infinity

Table 1: Scoring test trees