ULL 2015
Assignment 1
part 1

Introduction

In the last lecture you saw that EM is not particularly suitable to train the parameters of a context free grammar: although the likelihood of the corpus goes up during the training, the precision and recall go down (when the corpus is initialised with relative frequency estimation). In this assignment, you will create some graphs that illustrate this (as in lecture slide 31).

Assignment

You should start by choosing a treebank file, such as wsj01-21-without-tags-traces-punctuation-m40.txt. Extract a grammar from the treebank and then parse the sentences of the treebank with it (you can use only part of the treebank if you find training takes too long with the entire treebank). You should then use the program evalC to determine the precision and recall of this grammar. Determine also the likelihood of the corpus under this grammar.

Now that you have established the starting point of the grammar, start training the parameters of the grammar with the inside-outside implementation of Bitpar (as in the previous exercise). After every (couple) rounds, parse the sentences of the corpus and determine the accuracy and precision of the parses and compute the likelihood of the corpus under the current grammar. Plot the results in a graph.

If you finish early: try to reproduce the graph on slide 24.

Downloading and running evalC

You can evaluate your parses using the program evalC, that computes the recall and precision of your parses. First, download evalC into your working directory:

```
wget homepages.inf.ed.ac.uk/fsangati/evalC_25_5_10.zip
```

Extract the files. You can then run the program with a file with gold parses ($treebank_gold), your parses ($your_parses) and an output file ($output) to write the output of the program to:

```
java -jar evalC.jar $treebank_gold $your_parses $output
```

The precision, recall and F-score of the parse will appear in the terminal, more information can be found in the file $output.

Determining the likelihood of a corpus

To find the likelihood of a single sentence in the corpus, call bitpar with the -vp option. Bitpar will give you the probability of the TOP node in the parse forest (can you find it?), which equals the likelihood of the sentence under the grammar. If you print the parse forests of all sentences to a file called $output, you can get the likelihoods of the sentences with the following command:

```
cat $output | grep -E -o '^[{]*\(TOP=\#i |[P=[0-9]+|.[0-9]*[-e]*[0-9]* |]' | grep
-E -o '[0-9]+\.[0-9]*[-e]*[0-9]*'
```

You can then compute the likelihood of the entire corpus by multiplying all these probabilities (or rather, sum their logs to get the log probability). If you have trouble doing that, you can use the provided python script.