# Natural Language Processing:

## Assignment 6: Hooking up words and phrases and clauses

Jordan Boyd-Graber

Out: **16. September 2014**
Due: **17. October 2014**

## Introduction

As always, check out the Github repository with the course homework templates:

`git://github.com/ezubaric/cl1-hw.git`

The code for this homework is in the `hw6` directory.

## Problem 1: Treebank Probabilities (5 points)

To warm up, you'll first compute PCFG probabilities from data. You'll create a class that can read in sentences (through the `add_sentence` function and then answer queries using the `query` function.

Add the required functionality to the file `treebank.py`. You can see how the code will be called in Listing 1.

## Problem 2: Shift-Reduce Parsers (15 points)

Next, you'll create sequences of shift-reduce actions that can produce dependency parses from a string of words. You'll implement the `transition_sequence` function in the `oracle.py` file. Your code will be much simpler if you use generators.

An example of the shift-reduce sequence is given in Listing 2.

## Problem 3: Eisner Algorithm (40 points)

The goal of this section is for you to build code that can, given a parsing score function and a sentence, discovers the best untyped dependency tree that explains that sentence.

Listing 1: Estimating production probabilities from Treebank data

```
>>> tb_probs = PcfgEstimator()
>>> from nltk.corpus import treebank
>>> for ii in treebank.parsed_sents():
... tb_probs.add_sentence(ii)
...
>>> tb_probs.query('NN', 'man')
0.0009114385538508279
```

Youll also need some data files that are too big to put in Github. Please download them here:

- tb_counts.words
- tb_counts.tag

Were using a very simple score function built from the Dependencies over Time corpus. It does not use information about the direction or the part of speech of the corpus; it only uses the words involved (which is the head and which is the dependent). You can assume that all the values are log probabilities (i.e., less than or equal to zero).

You are responsible for completing the following functions in the `EisnerParser` class:

- `initialize_chart`: Create a chart that you can fill in later as you parse a sentence. Start with the base case of dynamic programming, giving single spans a score of 0.0.
- `fill_chart`: After your chart has been initialized, this function should fill in all the cells in the chart with their appropriate values (this is the hardest part!).
- `reconstruct`: Given the coordinates of a span, return an iterator over all of the edges that are part of the best parse of that span. This will require you to store backpointers in your `fill_chart` function.

## Extra Credit

Extra credit is available by providing a custom score function with the function `custom_sf` that performs better than the score function I provided. You

Listing 2: Creating a shift-reduce sequence from a dependency parse

```
>>> from nltk.corpus import treebank
>>> print(" ".join(conll2007.sents('esp.train')[1]))
El Banco_Central_Europeo – BCE – fijo el cambio oficial
                        del euro en los 0,9355_dolares .
>>> s = conll2007.parsed_sents('esp.train')[1]
>>> for x in transition_sequence(s):
... print(x.pretty_print(s))
...
s
s
l (Banco_Central_Europeo, El)
s
s
l (BCE, –)
s
r (BCE, –)
r (Banco_Central_Europeo, BCE)
s
l (fijo, Banco_Central_Europeo)
s
s
l (cambio, el)
s
r (cambio, oficial)
s
s
r (del, euro)
r (cambio, del)
r (fijo, cambio)
s
s
s
l (0,9355_dlares, los)
r (en, 0,9355_dolares)
r (fijo, en)
s
r (fijo, .)
r (None, fijo)
s
```

can store information (up to 1MB) in the optional upload `sf.dat`, which you can use in the `custom_sf` function).

- You will get no points if you do not describe how the score function is computed
- You will get at most 3 points if you use the Treebank data
- You will get at most 10 points if you use data other than the Treebank data

## Turning in Your Assignment

As usual, submit your completed code to Moodle. Make sure:

- You do not change filenames or function names
- You do not change the API of the functions
- Make sure you pass the included unit tests
- Do not add `print` statements to the code you turn in
- Add your name as a comment to all files you turn in