Tree Language Modeling

CSCI 562 Assignment 5
Prof. Kevin Knight (knight@isi.edu) and Jonathan May(jonmay@isi.edu)
50 points
**** due at the beginning of class Thursday, November 2, 2006 ****

October 26, 2006

Part Zero (0 pts, nothing to turn in)

1. Download and install Tiburon from http://www.isi.edu/licensed-sw/tiburon. Alternatively, you can use a pre-installed version on aludra, though depending on your models this may not be ideal, as aludra's memory limitations are somewhat severe for Tiburon. If you do want to use the installed version, it is located at:

   `/auto/home-scf-22/csci562/tiburon`

   Be sure you have your JAVA_HOME variable pointing to the correct version of java. This can be done by issuing the command:

   `setenv JAVA_HOME /auto/home-scf-22/csci562/j2re1.4.2_12/`

   or

   `export JAVA_HOME=/auto/home-scf-22/csci562/j2re1.4.2_12/`

   depending on your shell.

2. Familiarize yourself with Tiburon by reading at least the first three sections of the tutorial, located at

   `http://www.isi.edu/licensed-sw/tiburon/tiburon-tutorial.pdf`

Part One (8 pts)

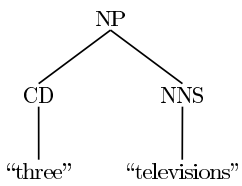
Turn in answers to the following questions **on paper**:
Consider the following RTGs:

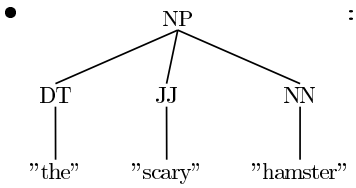
- ```
  %% filename: rtg1 %%
  q1
  q1 -> A(q1 q2)
  q1 -> A(q2 q1)
  q1 -> B(q2)
  q2 -> A(q2 q2)
  q2 -> A(q1 q1)
  q2 -> B(q1)
  q2 -> C
  ```
- ```
  %% filename: rtg2 %%
  qa
  qa -> A(qb qc)
  qa -> A(qc qc)
  qa -> A(qc qb)
  qb -> B(qc)
  qc -> C
  ```

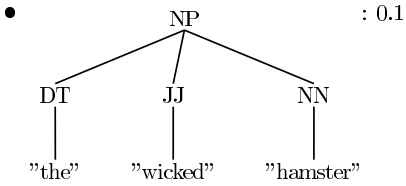
1. Describe, in plain English, the tree language represented by `rtg1`.
2. Write each of the trees in the language represented by `rtg2`. Which of these is also accepted by `rtg1`?
3. Write an RTG that represents the intersection of `rtg1` and `rtg2`, that is, that represents those trees accepted by both languages. Use Tiburon to accomplish the same task, and turn in the results.

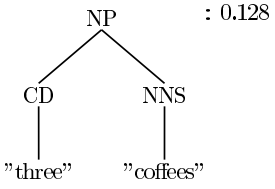
Part Two (7 pts)

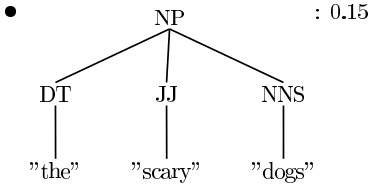
The following trees are the entire language represented by a WRTG:

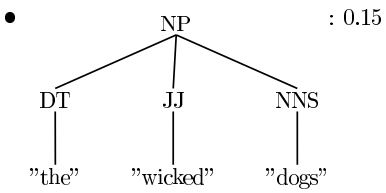
- 

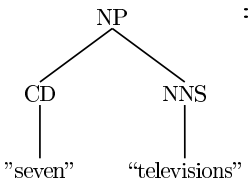
- 

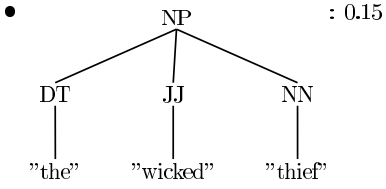
- 

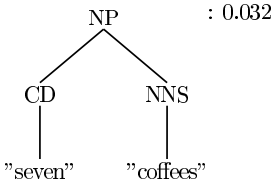
- 

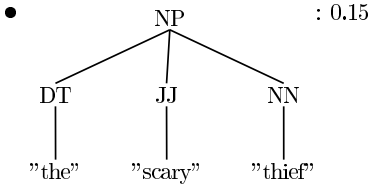
- 

- 

- 

- 

- 

- 

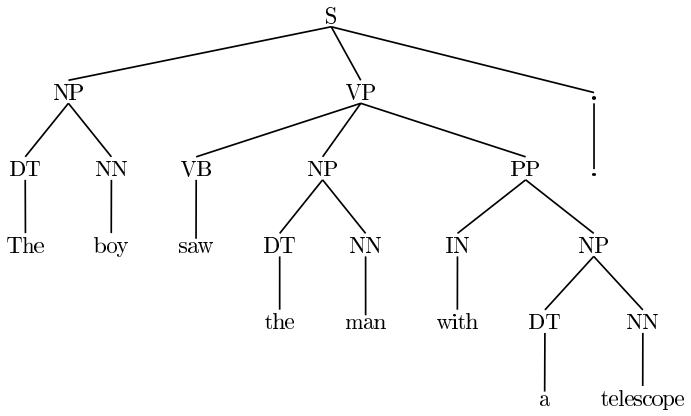
And here is *part* of the WRTG:

- ```
  %% filename: wrtg1 %%
  %% WARNING: this is only PART of a WRTG %%
  %% additional rules must be added!! %%
  q
  q -> NP(qdt qjj qnn) # .5
  qnn -> NN(‘‘hamster’’) # .4
  qcd -> CD(‘‘seven’’) # .2
  ```

Using this information, please complete the file `wrtg1` and print the entire `wrtg1` on paper. Remember, the language of the RTG should be these ten trees and no others, and the weight of each tree should be as indicated above. You can check your answer using `tiburon -c` and `tiburon -k`.

Part Three (35 pts)

In this part you will apply your language modeling skills honed in Assignment 2 to tree-based, rather than string-based languages. We can annotate English sentences with hierarchical *syntactic trees* to describe the function of sentence parts that are greater than one word in length. Here is an example of a sentence annotated with a syntactic tree:


  ```

The annotation tells us that the entire sentence (represented by the symbol "S") is composed of two substructures, a noun phrase (NP) and a verb phrase (VP). The noun phrase is composed of a determiner (DT) and a noun (NN), which are aligned to the words "the" and "boy", respectively. Modeling well-structured trees can be more effective than modeling n-gram phrases, especially in long sentences.

Task

Build a WRTG that is a language model of syntactic trees (just the trees, without the sentences they annotate). Your WRTG will assign probabilities to trees, giving commonly-seen trees higher probability, and less commonly-seen trees lower probability. A file of 5,000 sample training trees can be found in:

`/auto/home-scf-22/csci562/asst5/trees-train`

You will also find 100 "smoothing" trees in

`/auto/home-scf-22/csci562/asst5/trees-smooth`

and 100 sample "development test" trees in

`/auto/home-scf-22/csci562/asst5/trees-devtest`

You may assume that the test data will not contain any one-level productions not found in the training data. For your reference, the one-level productions (i.e. the CFG that represents the training data) can be found in the following file:

`/auto/home-scf-22/csci562/asst5/cfg`

Hints

As before, you should create a model that gives high probability to well-formed, common subtrees, and low probability to infrequently seen subtrees. There are lots of ways to use Tiburon to test the probability of various trees on your WRTG, but you may find the batch option, which calculates cross entropy of a corpus of trees, particularly useful. To evaluate your WRTG on the development test, do the following:

```
tiburon -b trees-devtest your-wrtg
```

You may see two warnings about "assuming ... is a batch file" and "using special cross-entropy batch mode". This is normal. Tiburon is slow compared to carmel - allow about 45 seconds on aludra for cross-entropy to be calculated. If you get warnings about not being able to get a derivation, this is an indication that your WRTG is faulty - you can try running `tiburon -bk 1 trees-devtest your-wrtg` – this will print each tree if the WRTG can recognize it, and will print a "0" if not. You can diagnose problems this way.

When you hand in your WRTG it will be tested on a secret set of sentences and **your score on this assignment will be based on the cross entropy of this test data**, so you will probably want to mimic these conditions as closely as you can. We provide an additional 100 sentences in:

`/auto/home-scf-22/csci562/asst5/trees-heldout`

Don't look at `trees-heldout` when you're testing and re-testing your WRTG (use `trees-devtest` for that); save it for a few evaluations. Other ways to test your WRTG include using the k-best (-k) and stochastic generation (-g) options of Tiburon, as described in the tutorial. You should also use the smoothing set to adjust your WRTG weights so they don't overfit the training data. Also, the normalization (-n) option ensures the probability mass will sum to one. Be sure to use it on your WRTG before submitting, as follows:

```
tiburon -no wrtg.normalized wrtg.notnormalized
```

This command takes as input the file `wrtg.notnormalized`, normalizes the weights (the `-n` flag), and outputs the file (the `-o` flag) as `wrtg.normalized`.

We've provided a couple of perl scripts that may aid you in your homework. You should think of these as motivation more than solution, as the best WRTG will require more information than these scripts can provide, but they are a good jumping-off point. Feel free to use them as you wish, modify them, ignore them, etc. Their comments provide documentation and annotation. Find them at:

`/auto/home-scf-22/csci562/asst5/extract-cfg.pl`
`/auto/home-scf-22/csci562/asst5/cfg2rtg.pl`

You may or may not wish to make use of Tiburon's EM training procedure. Given a file of trees, one per line, and an rtg file, the following procedure uses em to set weights on the rtg:

```
tiburon --randomize -o rtg.trained -t 5 training-file rtg.untrained
```

In the above command, `--randomize` randomly picks initial weights for the initial iteration (otherwise the weights on the untrained rtg are used), `-o` specifies where the output should be written, `-t 5` specifies that training is desired for 5 iterations (other numbers of course may be used), `training-file` is the file of trees used to train, and `rtg.untrained` is the input rtg. You may get a warning that looks like "Warning: RTG training assuming no counts for now...". Please ignore this. Also, note that training may take quite a long time, especially on aludra. You will see a cross-entropy statement after each iteration (the first may be inaccurate) to let you know it is still working.

What to turn in

- On paper, provide a description of your WRTG design, how you set the weights, in what ways the various data sets were used, and what type of smoothing was used, if any. Report the cross-entropy obtained on the file `trees-heldout` using the `tiburon -b` command. Stochastically generate 10 sample trees using the `tiburon -g` command.
- By email, submit your WRTG. Name the file *`email`*`.wrtg` where *email* is your email address before the domain. As an example, if I was submitting this homework my file would be named `jonmay.wrtg`.

A note on grading: As in previous assignments, the language model design portion is somewhat open-ended, in that those models that get the lowest cross-entropy on the test data will receive extra points on this homework, but there is no way of saying what the best possible score could be. (As a rough guide, all submissions should be able to get a cross entropy lower than 1.5 on the devtest data.) Additionally, well-motivated and interesting language model designs that, while not top-scoring, score competitively, will also be rewarded for creativity. Thus, please describe your strategy and motivation and comparisons with other strategies in good detail.

Overview Of This Homework

Below, we summarize what we expect you to hand in for this homework. For more complete details, see the relevant section of the homework above. Please only submit what is asked for below, in the format specified.

- **On paper**, the answers to questions 1a (2 pts), 1b (3 pts), and 1c (3 pts)
- **On paper**, your completed `wrtg1` from part 2 (7 pts)
- **On paper**, a description of your WRTG from part 3 (15 pts)
- **On paper**, the cross-entropy obtained from running your WRTG from part 3 on `trees-heldout` (5 pts)
- **On paper**, 10 stochastically generated sample trees from your WRTG (5 pts)
- **By email or other electronic means to jonmay@isi.edu**, your WRTG from part 3, named as the file *email*.wrtg, where *email* is your email address before the domain. If as a part of your discussion you want to compare and contrast different approaches to creating WRTGs and send these different versions, this is acceptable (note: this is not mandatory), but one of them *must* be labeled *email*.wrtg, and this is the one that will be considered for scoring. (10 pts)