Tree Transducers

CSCI 562 Assignment 6
Prof. Kevin Knight (knight@isi.edu) and Jonathan May(jonmay@isi.edu)
50 points
**** due at the beginning of class Thursday, November 16, 2006 ****

November 9, 2006

Part Zero (0 pts, nothing to turn in)

1. To aid you in this assignment you should use a recently updated version of tiburon (version 0.4.5). *Version 0.4.3 will not work!* You can use the one on aludra, which is located at:

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

   As before, 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. If you wish to use a personal copy, the latest version is available for your download at http://www.isi.edu/licensed-sw/tiburon.

2. Familiarize yourself with Tiburon's syntax for tree transducers by reading section four of the tutorial, located at

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

   For this assigment we will only be concerned with tree-to-tree transducers, so you don't need to worry about tree-to-string transducers.

Part One (10 pts)

Consider the following tree transducer:

```
q
q.A(x0: x1:) -> A(q.x1 r.x0)
q.B(x0:B) -> Z(q.x0)
q.B(x0:C) -> D(q.x0)
q.C -> F
r.A(x0: x1:) -> A(r.x0 E q.x1)
r.B(x0:) -> D(D(r.x0))
r.C -> D
```

As you can see, given the input tree B(C), this transducer would produce the output tree D(F). If the output tree from this transducer were A(F D), the input tree would be A(C C).

1. For each of the following input trees, what is the output tree produced from this transducer?
   (a) A(A(C C) C)
   (b) A(C A(C C))
   (c) B(B(B(B(C))))
2. For each of the following output trees, what is an input that could have produced this output from the transducer?
   (a) A(F D(D(D(D(D(D(D)))))))
   (b) A(F A(A(D(D(D)) E F) E Z(D(F))))

Part Two (40 pts)

Construct a tree-to-tree transducer that translates from English trees to Spanish trees. 10 translated sentences are available at:

`/auto/home-scf-22/csci562/asst6/trees`

The file is structured as follows:

```
engtree1
-----
spantree1

engtree2
-----
spantree2

...
```

where items above the line are English trees and items below the line are Spanish translations. If more than one tree is above or below the line, this indicates there is more than one possible translation of the sentence.

A vocabulary list is available at:

`/auto/home-scf-22/csci562/asst6/vocablist`

This file contains all the words you should be able to translate. You will notice that some words have multiple translations. These are generally due to the agreement properties of Spanish. We have provided a brief guide to Spanish grammar at:

`/auto/home-scf-22/csci562/asst6/dictionary`

This file discusses grammatical properties of Spanish words and includes translations of all of the words in the `vocablist` file, as well as some additional vocabulary words. Your transducer *must* be able to translate all of the words in the `vocablist` file — these are marked with an asterix (*) for your convenience in the `dictionary` file. You do *not* need to include translations in your grammar for words discussed in `dictionary` that are not in `vocablist`, though you can if you want to, for the purposes of creating new sentences, because you feel like a challenge, etc.

Your completed transducer should be able to translate all of the sentences in the `trees` file, and should be able to translate other simple sentences that contain the vocabulary of the `vocablist` file. You do *not* need to build a complete translation system — don't try to handle any sentences more complex than the ones in the `trees` file.

What to Turn In

- On paper, a description of the strategy used to build your transducer. Highlight interesting cases and discuss any problems you had. (5 pts)
- On paper, write 5 English sentence trees that are *not* in the `trees` file, and the Spanish translation or translations produced by your transducer. See below for how you can use Tiburon to do this. (5 pts)
- Could the approach you used be extended to translating general sentences of English? Discuss (on paper) the pros and cons of using such an approach. Give concrete examples. (10 pts)
- By email, submit your transducer. Name the file *`email`*`.trans` where *email* is your email address before the domain. As an example, if I was submitting this homework my file would be named `jonmay.trans`. (20 pts)

Hints

You will probably need to make use of *state* to complete this assignment. For example let's say you wanted to handle the following translations:

```
NP(DT(the) NN(boy))
-----
NP(DT(el) NN(chico))

NP(DT(the) NN(girl))
-----
NP(DT(la) NN(chica))
```

One solution would be as follows:

```
q
q.NP(x0:DT x1:NN) -> NP(q.x0 q.x1)
q.DT(the) -> DT(el)
q.NN(boy) -> NN(chico)
q.DT(the) -> DT(la)
q.NN(girl) -> NN(chica)
```

This seems okay, but it also generates these incorrect pairs:

```
NP(DT(the) NN(boy))
-----
NP(DT(la) NN(chico))

NP(DT(the) NN(girl))
-----
NP(DT(el) NN(chica))
```

You could take the brute force approach and simply recognize the whole phrase:

```
q
q.NP(DT(the) NN(boy)) -> NP(DT(el) NN(chico))
q.NP(DT(the) NN(girl)) -> NP(DT(la) NN(chica))
```

This solves the overgeneration problem, but you'd have to create a rule for each possible sentence, since you should be able to recognize this type of phrase with any noun in the vocabulary in the place of "boy" or "girl". You can use *state* to mandate the determiner agree with the noun as follows:

```
q
q.NP(x0:DT x1:NN) -> NP(qmasc.x0 qmasc.x1)
q.NP(x0:DT x1:NN) -> NP(qfem.x0 qfem.x1)
qmasc.DT(the) -> DT(el)
qfem.DT(the) -> DT(la)
qmasc.NN(boy) -> NN(chico)
qfem.NN(girl) -> NN(chica)
```

This transducer doesn't overgenerate, and if you want to recognize additional phrases, all you have to do is add the appropriate rules, like so:

```
qmasc.NN(dog) -> NN(perro)
qfem.NN(house) -> NN(casa)
```

And so on. You'll find state will help you with the other agreements required in Spanish that are detailed in the `dictionary` file.

Tiburon can help you complete this task. The `-l` switch passes trees on the left of your transducer, producing the resultant RTG. If you type a command like:

```
> echo ‘‘A(B(D))’’ | tiburon -ls demotrans
```

and the result has no rules in it, your transducer can't process the input tree and you probably have something wrong. If you use the `-k` switch you can output the trees recognized by the result rtg:

```
> echo ‘‘A(B(C))’’ | tiburon -lsk 3 demotrans
This is Tiburon, version 0.4.5
Warning: returning fewer trees than requested
F(E(D)): 1.0000
D(E(F)): 1.0000
0
```

This indicates there are two transformations of the input tree (we asked for three but only two could be found). If you use the `-b` batch mode switch, you can provide a list of input trees, and each will be processed in turn. For example:

```
> tiburon -blk 3 demobatch demotrans
```

will perform left application of each tree in `demobatch` to the transducer in `demotrans` and attempt to get the top 3 trees. While you construct your transducer, use these tools (or whatever else you can think of) to check that your transducer is doing what you expect.

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 the five questions in part 1 (2 pts each)
- **On paper**, a description of your transducer from part 2 (5 pts)
- **On paper**, 5 English sentence trees that did *not* appear in the `trees` file and the Spanish trees produced by your transducer for those trees. (5 pts)
- **On paper**, the answer to the question about extending the approach you used to general translation. (10 pts)
- **By email or other electronic means to jonmay@isi.edu**, your transducer from part 2, named as the file *email*.trans, where *email* is your email address before the domain. (20 pts)