Homework 1

CSCI 562

Due at 11am, 18 Sep 2011

CN Y RD NGLSH WTHT VWLS? In this assignment, you will build a finite-state machine to automatically restore vowels to vowelless text. (In English, this may not be a very useful thing to do, but in languages like Arabic and Hebrew where vowels are regularly not written, it is extremely useful.)

Before you begin

- Download and install Carmel (`http://www.isi.edu/licensed-sw/carmel`). You can find pre-compiled binaries for various platforms in the directory `graehl/carmel/bin`.
- Read Sections 1 and 2 of "A Primer on Finite-State Software for Natural Language Processing" (`http://www.isi.edu/licensed-sw/carmel/carmel-tutorial2.pdf`). A brief overview of some of Carmel's command-line options (run `carmel` to see complete list):

`-si`	expect a string on stdin
`-l`	compose stdin onto the left of the named FSA/FST(s)
`-r`	compose stdin onto the right of the named FSA/FST(s)
`-O`	print only output labels, suppress input labels
`-I`	print only input labels, suppress output labels
`-k` n	list k sequences rather than the whole FSA/FST
`-b`	batch mode: process multiple input lines
`-WE`	suppress weights and empty labels in k-best lists

- Download and unpack `hw1.tgz` from Blackboard. This contains:

`hw1.pdf`	this file
`vocab`	list of English words
`vocab.small`	shorter list (for testing purposes only)
`strings`	English sentences
`strings.bad`	English sentences with bad spelling
`eval.py`	script for measuring accuracy

Finite-state acceptors

1. For all the questions below, use `vocab`. The other file, `vocab.small`, is just for testing purposes.
   (a) How many words are there?
   (b) How many distinct characters (or: how many character *types*) are there?
   (c) How many character occurrences (or: how many character *tokens*) are there?

   For example, the word *banana* has 6 character tokens, but only 3 character types.

2. Create a FSA in Carmel format called `english.fsa` that accepts all strings consisting of English words (as defined by `vocab`) separated by underscores, and nothing else. It should be letter-based, not word-based: that is, transitions should be labeled with letters, not whole words. For example, it should accept:

   ```
   "T" "H" "I" "S" "_" "I" "S" _ "A" _ "S" "T" "R" "I" "N" "G"
   ```

   (a) How many states and how many transitions does your FSA have? (Use `carmel -c english.fsa`.) *You will be graded on how small your FSA is.*

   (b) Draw a diagram that shows how you created your FSA. You won't, of course, be able to draw the whole FSA, but your diagram must have enough detail for someone else to replicate your FSA.

   (c) Verify that your FSA accepts every line in `strings` and no lines in `strings.bad`. Show the output of Carmel on the first five lines of each file. You can use commands like:

   ```
   cat strings | carmel -slib english.fsa
   ```

Finite-state transducers

3. Create a FST in Carmel format called `remove-vowels.fst` that deletes all English vowels, preserving word boundary information. For the purposes of this assignment, vowels are defined to be members of {`A`, `E`, `I`, `O`, `U`}. For example, it should perform the following mappings:

   ```
   "Y" "O" "U" "_" "A" "R" "E" "_" "H" "E" "R" "E"  ↔  "Y" "_" "R" "_" "H" "R"
       "R" "E" "A" "D" "_" "A" "_" "B" "O" "O" "K"  ↔  "R" "D" "_" "_" "B" "K"
   ```

   (a) Draw your FST on paper in enough detail for someone else to replicate it.

   (b) Test your FST in the forward direction on `strings` with the following command:

   ```
   cat strings | carmel -slibOEWk 1 remove-vowels.fst
   ```

   Show the output on the first five lines, and save the whole output to a file `strings.novowels`.

   (c) Test your FST in the backward direction with the following command:

   ```
   echo '"B" "L" "D" "N" "G"' | carmel -sriIEWk 10 remove-vowels.fst
   ```

   The `-k 10` option asks for up to 10 output strings. How many output strings do you get? Why?

4. Now you can use backwards application of your FST to do vowel restoration.

   (a) Run your vowel restorer on `strings.novowels`, using the following command:

   ```
   cat strings.novowels | carmel -sribIEWk 1 remove-vowels.fst
   ```

   Show the output on the first five lines, and save the whole output to a file `strings.restored`.

   (b) Compute your vowel-restoration accuracy using the command:

   ```
   python eval.py strings strings.restored
   ```

   What was your accuracy?

   (c) Why is the score so low?

Combining FSAs and FSTs

5. The vowel restorer you built in the previous part had a problem. Can you fix it by combining your FSA and FST?
   (a) Describe how to combine your FSA and FST to improve vowel restoration.
   (b) Implement your idea and test it on `strings.novowels`. What is your new accuracy?
   (c) Are the results satisfactory? Why doesn't the machine do what a human would do?
6. How might your vowel restorer be further improved? Come up with at least one idea, and for each idea:
   (a) Describe it in enough detail so that someone else could replicate it.
   (b) Implement it and try it on `strings.novowels`.
   (c) Report your new accuracy on `strings.novowels`. *You will be graded on your final accuracy.*

What to turn in

On **paper**:

- Your answers to all questions above

On **Blackboard**:

- Your `english.fsa`, and the code that you used to generate it
- Your `remove-vowels.fst`, and (if any) the code that you used to generate it
- Any other code that you wrote in this assignment