Assignment 1
CS562
Prof. Kevin Knight
Due at the <u>beginning</u> of class Thursday, Sept 7, 2006

Please turn in answers on paper (& e-mail) at the beginning of class.
Late assignments receive 30% off.
All work must be your own – no collaborations on homeworks.
Please write in complete sentences and transmit your ideas clearly and fully.

==========

0. Warm-Up (No Credit)

Get experience with the Carmel finite-state software package by:

a) Downloading and installing Carmel, and locating the command-line binary "carmel". On aludra.usc.edu, a Carmel binary is available at /auto/home-scf-22/csci562/carmel/bin/carmel.

b) Reading 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) and issuing the commands referred to there.

c) Testing the software on any of the finite-state acceptors (FSA) and transducers (FST) covered in class.

==========

1. Finite-state acceptors (FSA)

Create and test a finite-state acceptor that accepts valid English letter strings. We define a valid string as one composed of words from

```
/auto/home-scf-22/csci562/asst1/vocab
```

If a string has more than one word, the words should be separated by the underscore character "_", e.g.:

```
R E A D _ A _ B O O K
```

NOTE: The vocab corpus is very large, so you should test your FSA on a small corpus first, such as vocab.small.

- **How many words are there in the vocab file?**
- **How many distinct characters are there?**

a. Create a *concise* Carmel FSA called english.fsa that accepts any valid English letter string. It should reject any string containing a word not in vocab. The FSA should be "letter-based", and

not “word-based” (transitions should be labeled with letters, such as “C”, not with whole words, such as “COMPANY”). *You will be graded on how concise the FSA is*. You can get the size by typing:

```
carmel –c english.fsa
```

- **Turn in one page worth of printed transitions from this FSA.**
- **Sketch on paper a drawing of your FSA scheme in enough detail for someone to duplicate it. Turn in this drawing.**
- **Send a pointer to your FSA file to jonmay@isi.edu, or the whole file if a pointer is not possible.**
- **How big is your FSA in states and transitions?**

b. Find two letter sequences, one that your FSA rejects and one that it accepts. To see if the sentences in your FSA, you may use commands like:

```
echo ‘ “R” “E” “A” “D” “_” “A” “_” “B” “O” “O” “K” ‘ | carmel -sli english.fsa
echo ‘ “R” “A” “E” “D” “_” “A” “_” “B” “I” “I” “K” ‘ | carmel -sli english.fsa
```

- **Show results of these commands.**

2. Finite-state transducers (FST)

Create and test a Carmel transducer called remove-vowels.fst that deletes English vowel letters A, E, I , O, and U. Here are sample mappings that your FST should encode:

```
“C” “A” “N” “_” “Y” “O” “U” “_” “R” “E” “A” “D” “_” “T” “H” “I” “S” <=>
        “C” “N” “_” “Y” “_” “R” “D” “_” “T” “H” “S”

“Y” “O” “U” “_” “A” “R” “E” “_” “H” “E” “R” “E” <=>
        “Y” “_” “R” “_” “H” “R”

“I” “_” “A” “M” “_” “H” “E” “R” “E” <=>
        “_” “M” “_” “H” “R”

“R” “E” “A” “D” “_” “A” “_” “B” “O” “O” “K” <=>
        “R” “D” “_” “_” “B” “K”
```

Note that vowel deletion should preserve word boundary information.

a. Draw your FST on paper in enough detail to permit someone to duplicate it.

- **Turn in this drawing.**

b. Test your FST in the forward direction with the following command:

```
echo ' “B” “U” “I” “L” “D” “I” “N” “G” ' | carmel -sliOEWk 10 remove-vowels.fst
```

This should return the sequence with vowels deleted.

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

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

- **Turn in these commands and the results.**
- **How many resulting strings do you get when you apply the FST in the forward direction? Why?**
- **How many resulting strings do you get when you apply the FST in the backward direction? Why?**

NOTE: carmel usage notes
-si = expect a string on standard input, as supplied with echo
-l = compose the standard input onto the left of the named FSA/FST(s).
-r = compose the standard input 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 out k sequences rather than the whole FSA/FST.
-WEQ = suppress weights, empty labels, quote marks in top-k lists
type carmel for more switches.

3. Combining FSA and FST

A problem occurs when you use your FST in the backward direction. Can you alleviate the problem by combining your FSA and FST?

- **Turn in a description of your idea and the results you got, including quantitative measures of how many strings get produced at various stages, for various inputs including:**

  “F” “_” “Y” “_” “C” “N” “_” “R” “D” “_” “T” “H” “S”

- **Are the results satisfactory? Why doesn’t the machine do what a human decoder would do?**

- **How might the FSA and/or FST be further improved?**

4. No Word Boundaries

Modify your FSA and FST to work on strings that have no word boundaries explicitly marked, e.g.:

“R” “D” “T” “H” “S”

instead of

"R" "D" "_" "T" "H" "S"

Your machines should restore word boundaries as well as vowels.

- **What changes did you make to your FSA and/or FST?**
- **What happens?**
- **Turn in your findings with examples and quantitative measures.**

Optional assignment (no credit, only if you want to learn more)

Write a Carmel FST that transforms English letter sequences into English phoneme sequences, (and vice-versa when applied in reverse). It should be able to pronounce the words in vocab, but *also new words it has never seen before*.

Test your FST on a list of ten words total, five words drawn randomly from the file:

/auto/home-scf-22/csci562/asst1/word-pron

and five words drawn from outside this list, such as names or technical terms. You can use the file word-pron in FST development.