Homework #1

CS662, Fall 2017
Individual work only, no collaboration
Due on Blackboard **Thursday, September 7, 11:59pm**
*** Note two-day extension to Thursday because of 3-day weekend ***
Late homework 30% off (up to one week late)

Here are some interesting facts about language:

- Because Arabic is written without vowels, speech synthesis (automatically converting text to speech) is a challenge. One approach is to "re-vowelize" Arabic text before synthesizing it.
- Chinese is written without spaces between words. A pre-processor that inserts spaces is a great help for downstream processes like machine translation.
- People frequently make typographical errors, which must be corrected before other processing is applied.
- Speech recognition systems must convert sounds into words, but distinct words may have the same sounds, and distinct sound sequences may be spelled the same.

This assignment investigates simplified versions of these tasks, all using Spanish data.

0. Warm-up

- Download the Carmel software package (https://github.com/isi-nlp/carmel)
- Read *A Primer on Finite-State Software for Natural Language Processing*
- Test Carmel on some of the finite-state acceptors (FSAs) and transducers (FSTs) covered in class
- Download the file *spanish-vocab.txt* from Piazza. This file contains over 90,000 distinct Spanish words (plus some noise, including English words and names). Note that accents and tildes have been removed to simplify processing.

1. Create and test a concise finite-state acceptor (FSA) that accepts letter strings consisting of words from *spanish-vocab.txt*, and rejects all others. Call your file *spanish.fsa*. If a string has more than one word, words are separated by the underscore character `"_"`. For example, your FSA should accept this string:

```
L O S _ G A T O S _ D U E R M E N
```

Your FSA should be "letter-based", and not "word-based" (transitions labeled with letters, such as "G", not with whole words, such as "GATO"). The vocabulary file is large, so you should work with a portion of the file at first. You can test your FSA with commands like this:

```
echo ' "L" "O" "S" "_" "G" "A" "T" "O" "S" ' | carmel -sli spanish.fsa
echo ' "L" "O" "S" "G" "_" "R" "I" "S" "T" ' | carmel -sli spanish.fsa
```

You should make the FSA as small as you can, but you will not be graded on the size, only on correctness. You can get the size by typing:

```
carmel -c spanish.fsa
```

2. Build an FST called *vowel-deleter.fst*, which removes all occurrences of A, E, I, O, and U from a string.

Test it in the forward direction, e.g.:

```
echo ' "G" "A" "T" "O" "_" "A" "Q" "U" "I" ' | carmel -sliOEWk 10 vowel-deleter.fst
```

Test it in the backward direction, e.g.:

```
echo ' "P" "R" "R" "_" "Q" ' | carmel -sriIEWk 10 vowel-deleter.fst
```

Carmel usage:

-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. Build an FST called *space-deleter.fst*, which removes all occurrences of `"_"` from a string.

Test it in the forward and backward directions.

4. Build an FST called *typo.fst*, which introduces typographical errors into a string. It should replace every third letter with a mistake (spaces do not count as letters for this purpose). The mistake should result from hitting a neighboring key on the keyboard, e.g., R might be replaced with E, D, F, or T.

Test it in the forward and backward directions.

5. Build an FST called *spell.fst*, which converts a Spanish phoneme (sound) sequence into a Spanish letter sequence. Use the following phoneme inventory:

pho-a, pho-b, pho-ch, pho-d, pho-e, pho-f, pho-g, pho-h, pho-i, pho-k, pho-l, pho-m, pho-n, pho-o, pho-p, pho-r, pho-rr, pho-s, pho-t, pho-th, pho-u, pho-y

and the following spell-out rules:

pho-a -> A
pho-b -> B *or* V
pho-ch -> C H
pho-d -> D
pho-e -> E
pho-f -> F
pho-g -> G
pho-h -> J
pho-i -> I
pho-k *(when followed by pho-e or pho-i)* -> Q U
pho-k *(when followed by pho-s)* -> X
pho-k *(when followed by pho-a, pho-l, pho-o, pho-r, or pho-u)* -> C
pho-l -> L
pho-m -> M
pho-n -> N
pho-o -> O

pho-p -> P
pho-r -> R
pho-rr -> R R
pho-s *(when preceded by pho-k)* -> nothing
pho-s *(when not preceded by pho-k*) -> S
pho-t -> T
pho-th *(when followed by pho-a, pho-o, or pho-u)* -> Z
pho-th *(when followed by pho-e or pho-i)* -> C or Z
pho-u -> U
pho-y -> Y *or* L L
nothing -> H   *(NOTE: letter h is silent in Spanish.)*

Test it in the forward direction, e.g.:

```
echo ' "pho-k" "pho-a" "pho-y" "pho-e" ' | carmel -OQWEslik 5 spell.fst
```

Test it in the backward direction, e.g.:

```
echo ' "C" "A" "L" "L" "E" ' | carmel -IQWEsrik 5 spell.fst
```

6. Use *spanish.fsa* to improve the results of vowel restoration (backward application of *vowel-deleter.fst*), space insertion (backward application of *space-deleter.fst*), typo correction (backward application of *typo.fst*), and phoneme-to-text conversion (forward application of *spell.fst*).

===

WHAT TO TURN IN

Create a write-up (*writeup.txt*) that describes the results of each of your experiments and what you learned. *Your write-up will be graded on how clear and insightful it is.* Sketch your FSA and FSTs in such a way that someone could duplicate them. Include specific Carmel inputs and outputs to support your points.

Create and upload (to Blackboard) a single tar file containing *writeup.txt*, *spanish.fsa*, *vowel-deleter.fst*, *space-deleter.fst*, *typo.fst*, and *spell.fst*.

Your tar file should be named *FirstName_LastName_Assignment1* (e.g., *John_Smith_Assignment1*).

===

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 vocabulary, 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: **http://svn.code.sf.net/p/cmusphinx/code/trunk/cmudict/cmudict.0.7a**, plus five words drawn from outside this list, such as names or technical terms.