Homework #2 CS662, Fall 2017 Prof. Kevin Knight

Due **Tuesday, September 19**, midnight.
Individual work, no collaborating on solutions.
Late homework 30% off (up to one week late).
Questions to Piazza.

==

NOTES ON CARMEL SWITCHES

==

```
-k n   return top-n answers

-s     input is coming via standard input, as in the “echo” commands above
-i     input is coming in as a string of symbols, not as a WFSA

-l     incoming string should be put on the left side of the transducer
-O     print what comes out of the right (output) side of the transducer

-r     incoming string should be put on the right side of the transducer
-I     print what comes out of the left (input) side of the transducer

-EQ    suppress *e* and quote marks when printing result strings

if you have –l, you usually want –O
if you have –r, you usually want -I
```

1. Transliteration

In Japanese text, Western names and technical terms are often transliterated into special symbols called *katakana*. Here are some examples:

Angela Johnson アンジラ・ジョンソン (a n jira jyo n so n)	*New York Times* ニューヨーク・タイムズ (nyu u yo o ku ta i mu zu)	*ice cream* アイスクリーム (a i su ku ri i mu)	
Omaha Beach オマハビーチ (omahabiitchi)	*pro soccer* プロサッカー (purosakkaa)	*Tonya Harding* トーニャ・ハーディング (toonya haadingu)	
ramp ランプ (ranpu)	*lamp* ランプ (ranpu)	*casual fashion* カジュアルヒァッション (kajyuaruhasshyon)	*team leader* チームリーダー (chiimuriidaa)

Katakana is a syllabary, in which single symbols stand for whole syllable sounds (such as “ko” or “ra”). Because spoken Japanese consists largely of consonant-vowel syllables, a single katakana symbol often stands for two Japanese phonemes.

When we observe a katakana string, such as ナイト, we want to translate it into English. We imagine the katakana string arrived via this generative, noisy-channel process:

$\boxed{\text{P(eword)}}$ → English word sequence (e.g., Knight, 1 token) →

$\boxed{\text{P(epron | eword)}}$ → English phoneme sequence (e.g., N AY T, 3 tokens) →

$\boxed{\text{P(jpron | epron)}}$ → Japanese phoneme sequence (e.g., N A I T O, 5 tokens) →

$\boxed{\text{P(katakana | jpron)}}$ → Katakana (e.g., ナ イ ト, 3 tokens)

We'll focus on the first three steps of the model. Our ultimate goal is therefore to take Japanese phoneme sequences like these:

```
A N J I R A N A I T O
S U CH I I B E N R A R U Z U
D O N A R U D O T O R A N P U
SH Y E R I R U S A N D O B A A G U
```

and decode them back into English. You are given:

eword.wfsa a unigram WFSA of English word sequences
eword-epron.wfst a WFST from English words to English phoneme sequences
epron-jpron.data a database of aligned English/Japanese phoneme sequence pairs

(These and all other files for this homework are provided on Piazza as **hw_provided.tar.gz**, a gzipped tar file).

Play with the first two models, using commands like these:

```
% echo '"DATA"' | carmel -sliOEQk 5 eword-epron.wfst
% echo '"N" "AY" "T"' | carmel -sriIEQk 5 eword-epron.wfst
% echo '"JOHNSTON"' | carmel -sriIEQk 5 eword.wfsa
% echo '"JOHN" "STUN"' | carmel -sriIEQk 5 eword.wfsa
% echo '"JAW" "STUN"' | carmel -sriIEQk 5 eword.wfsa
% echo '"WHALEBONES"' | carmel -sliOEQk 5 eword-epron.wfst
```

Question 1. What commands did you try, and what did you learn about these machines?

Use **eword.wfsa** and **eword-epron.wfst** together to answer this question:

Question 2. What are the five most frequent words with the pronunciation "B" "EH" "R", and what are their probabilities? Show your command and Carmel output.

Create a machine called **epron-eword.wfst**, the inverse of **eword-epron.wfst**:

```
% carmel -v eword-epron.wfst > epron-eword.wfst
```

(Note that this command may take some time. You do not need to turn in this new machine on Blackboard.) Now use both **eword-epron.fst** and **epron-eword.wfst** to answer this question:

Question 3. What words sound the same as the word WHERE? What word sequences sound the same as the sequence ICE CREAM? Show your commands and Carmel output.

Finally, let's decode Japanese, by first building **epron-jpron.wfst** from dictionary data. The file **epron-jpron.data** consists of string pairs, English phoneme sequences paired with Japanese phoneme sequences. For each pair, Japanese phonemes are assigned integers telling which English phonemes align to them. For example:

```
"AE" "K" "T" "ER"              ;; English phoneme sequence for "actor"
"A" "K" "U" "T" "A" "A"        ;; Same word, loaned into Japanese
 1   2   2   3   4   4         ;; E.g., Japanese "T" maps to 3rd English sound
```

From the data, we can see that each English phoneme maps to one or more Japanese phonemes. Using this data, create a WFST called **epron-jpron.wfst**. It should probabilistically map English phoneme sequences onto Japanese ones, behaving something like this in the forward direction:

```
% echo '"L" "AE" "M" "P"' | carmel -sliOEQk 5 epron-jpron.wfst
R A M P
R U A M P
R A M U P
R A M P U
R A M PP U
  etc.
```

You need not include transitions with probability < 0.01.

Now use **eword.wfsa**, **eword-epron.wfst**, and **epron-jpron.wfst** on the same Carmel command line to decode the following Japanese sequences into English (with "-k 5" option). You can find these in **japanese.txt**.

```
"A" "N" "J" "I" "R" "A" "N" "A" "I" "T" "O"
"S" "U" "CH" "I" "I" "B" "E" "N" "R" "A" "R" "U" "Z" "U"
"D" "O" "N" "A" "R" "U" "D" "O" "T" "O" "R" "A" "N" "P" "U"
"SH" "Y" "E" "R" "I" "R" "U" "S" "A" "N" "D" "O" "B" "A" "A" "G" "U"
```

Question 4. What results did you get? Show your commands and Carmel output.

2. Part of speech tagging

In this part of the assignment, you will write a program to automatically tag sequences of words with their parts of speech (noun, verb, determiner, etc.). Training data is provided in the file **train-data**. Below, w=word, t=tag.

Build a P(w…w | t…t) channel model in Carmel WFST format. Call it **tag-to-word.wfst**. This model should be context independent, i.e.,

$$P(w\ldots w \mid t\ldots t) = P(w1 \mid t1) * P(w2 \mid t2) * \ldots * P(wn \mid tn).$$

Build two P(t…t) tag language models -- unigram and bigram -- in Carmel WFSA format. Call them **unigram.wfsa** and **bigram.wfsa**.

Use Carmel to compute optimal tag sequences using (1) the unigram model + channel model, and (2) the bigram model + channel model. Test the two tagging systems on the sentences in **test-data-1.sent** and turn in the results.

% carmel -brIEQk 5 unigram.wfsa tag-to-word.wfst test-data-1.sent
% carmel -brIEQk 5 bigram.wfsa tag-to-word.wfst test-data-1.sent

Correct answers are in **test-data-1.correct**. Investigate the errors made by your systems.

Question 5: How many errors did each system make?

Question 6. What do you attribute the errors in the bigram-based system to? Poor modeling of grammar, poor tag-to-word probabilities, sparse data ... ?

Question 7. What might you do to correct those errors?

Implement your own Viterbi decoder for the bigram system. Put your code in a directory called **viterbi/** and include a **README** file that tells us how we can run it. Tag the test sentences using this decoder instead of Carmel.

Question 8. Show a trace of your program running on the first sentence as it fills in the first two columns of Viterbi cells with Q[i,j] values.

Question 9. Over all three sentences, what percentage of your Viterbi cells get non-zero Q[i,j] values?

Question 10. Do you get the same output tag sequences as with Carmel?

3. Code-breaking

You intercept the following piece of enciphered English:

```
ciphertext A:  er vsm gtrrxr upit vst imyoa dvormvr fodvpbrtd s esu yp trqsot oy
```

Fortunately, you receive a videotape of an adversary enciphering another message using the same code system:

```
plaintext B:   a quick brown fox jumps over the lazy dog
ciphertext B:  s wiovl ntpem gpc kizqd pbrt yjr asxu fph
```

Build a Carmel WFST called **decipher.wfst** using this data and apply your WFST to ciphertext A.

Question 11. What decipherment string do you get?

Data for this part of the assignment is in the file **code.txt**

==

SUMMARY OF FILES PROVIDED

==

eword.wfsa	a unigram WFSA of English word sequences
eword-epron.wfst	a WFST from English words to English phoneme sequences
epron-jpron.data	a database of aligned English/Japanese phoneme sequence pairs

japanese.txt	four Japanese phoneme test strings ("really English, but written…")
train-data	198,796 English word tokens manually tagged for part-of-speech
test-data-1.sent	three English strings
test-data-1.correct	correct manual tagging for those three English strings
code.txt	two cipher strings, plus correct plaintext for the second string

===

WHAT TO TURN IN ON BLACKBOARD

===

Make a tar file called FirstName_LastName_Assignment2.tar. It should unpack *into a directory* called FirstName_LastName_Assignment2/. That directory should include:

writeup.{txt, pdf}	your answers to questions
epron-jpron.wfst	transducer from English phoneme sequences to Japanese phoneme sequences
tag-to-word.wfst	transducer from tag sequences to word sequences
unigram.wfsa	unigram tag model
bigram.wfsa	bigram tag model
viterbi/ (including README)	Viterbi decoding program
decipher.wfst	transducer from plaintext to ciphertext