Assignment 4
CSCI562 (Prof. Kevin Knight)
50 points
Due at the beginning of class Thursday, Oct 19, 2006

The topic of this homework is translating names and borrowed words across languages like English and Japanese. For example, the English word "computer" translates into Japanese characters that sound like "konpyuutaa" or perhaps "konpyuuta." The name "Angela Knight" becomes Japanese characters that sound like "anjiranaito."

We can hypothesize a four-stage generative model of this process, each stage implemented by a finite-state transducer:

1. An English phrase w is produced with probability P(w).
   => KNIGHT
2. The phrase w is converted to a phoneme sequence e with probability P(e | w).
   KNIGHT => N AY T
3. The English phoneme sequence e is translated to a Japanese phoneme sequence j with probability P(j | e).
   N AY T => N A I T O
4. The phoneme sequence j is rendered in Japanese syllabic characters k with probability P(k | j).

N A I T O => ナ イ ト
(na i to)

We will ignore the fourth stage here, and deal only with Japanese phoneme sequences like N A I T O.

The files

```
/auto/home-scf-22/csci562/asst4/w.wfsa
/auto/home-scf-22/csci562/asst4/w-e.wfst
```

implement stages 1 and 2 above.

This assignment is concerned with stage 3. The file

```
/auto/home-scf-22/csci562/asst4/ej.data
```

contains a list of over 2500 English/Japanese word pairs, in phonetic format. This can be used to train a transducer for stage 3. Here is one such word pair:

```
AH L ER T       (English sounds representing the word "alert")
A R A A T O     (Corresponding Japanese sounds)
```

In this homework, we assume a very simple fleshed-out generative story for stage 3 (how English sound sequences get probabilistically converted into Japanese sound sequences):

Given: an English sound sequence $e_1 \ldots e_r$.
Process: for i = 1 to r
  output j...j (a particular sequence of *one or more* Japanese sounds)
  with probability P(j...j | e_i)

A sample entry in the P(j...j | e) table might be as follows:

P(T O | T) = 0.42 /* chance that English sound "T" translates to Japanese sounds "T" "O" */

An *alignment* of a word pair consists of connections between the English sounds and Japanese sounds. Following the generative story,

- each English sound may be connected to one or more (contiguous) Japanese sounds,
- each Japanese sound must be connected to exactly one English sound, and
- connections may not cross each other

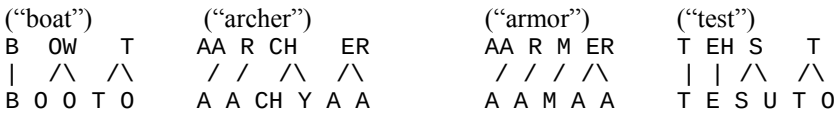
These are legal alignments:

These are not legal alignments:

[Note that the generative story requires that the English side have *not more sounds* than the Japanese side.]

Part I. Complete data, incomplete model (5 points)

Given the following manually aligned word pairs:



Calculate (and hand in) maximum likelihood estimates for:

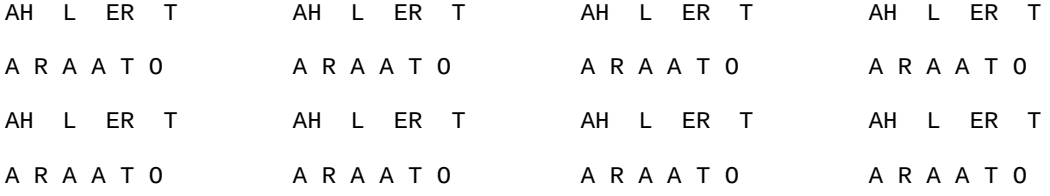
P(A A | AA)	= ?	P(T | T)	= ?
P(M | AA)	= ?	P(T O | T)	= ?
P(A | AA)	= ?	P(S | T)	= ?

Part II. Complete model, incomplete data (5 points)

Given the following model parameters:

P(A | AH)	= 0.8	P(A T O | T)	= 0.0
P(A R | AH)	= 0.1	P(T O | T)	= 0.5
P(A R A | AH)	= 0.1	P(O | T)	= 0.1
P(R | L)	= 0.6	P(A | ER)	= 0.1
P(R A | L)	= 0.2	P(A A | ER)	= 0.6
P(R A A | L)	= 0.1	P(A A T | ER)	= 0.0
P(A | L)	= 0.0	P(A T | ER)	= 0.0
P(A A | L)	= 0.0	P(T | ER)	= 0.0

Draw (and turn in) all of the legal alignments for the pair AH L ER T / A R A A T O. Circle the most probable alignment according to the model parameters above:



```
AH  L  ER  T        AH  L  ER  T

A R A A T O         A R A A T O
```

Part III. Incomplete model, incomplete data

(15 points)

Given the unaligned word pairs in the file

```
/auto/home-scf-22/csci562/asst4/ej.data
```

Write a program to generate all legal alignments for any word pair, such as those found in this file. Print out (and turn in) all legal alignments for the first five word pairs, and turn in your code listing.

(20 points)

Implement an EM algorithm for computing parameter estimates and best word-pair alignments for all word pairs in the above file. At each EM iteration, record the highest-scoring alignments for each of the first 20 word pairs. Turn in these alignments and a listing of your program.

(5 points)

Format your learned parameter estimates into a Carmel weighted finite-state transducer called `e-j.wfst`. Only include transitions with probability bigger than 0.01.

Use Carmel as a decoder to come up with proposed English translations for Japanese inputs in the file test-data:

```
/auto/home-scf-22/csci562/asst4/test-data
```

with contents:

```
"D" "A" "N" "I" "E" "R" "U" "H" "I" "Y" "U" "U" "R" "E" "TT" "O"
"U" "E" "S" "U" "R" "I" "I" "K" "A" "A"
"A" "N" "J" "I" "R" "A" "N" "A" "I" "T" "O"
"J" "Y" "O" "N" "A" "S" "A" "N" "M" "E" "E"
```

Use this command:

```
% carmel -brIEQk 5 w.wfsa w-e.wfst e-j.wfst test-data
```

Turn in the results of this command, plus your file `e-j.wfst`.

NOTE FOR THIS ASSIGNMENT: Turn in all work on paper only, with the exception of the WFST in the final section, which you should also send by email to `jonmay@isi.edu`.

Hint. A possible algorithm for enumerating the set of legal alignments for a given word pair

/* function **align** returns a complete set of alignments, requires that r <= m */

function **align**($e_1...e_r$, $j_1...j_m$)
if (r == 1) then
  return set consisting of a single alignment: { ((e_1 <-> $j_1...j_m$)) }
else
  result = empty set of alignments
  for i = 1 to m - r + 1
    p = **align**($e_2...e_r$, $j_{i+1}...j_m$)
    to each alignment in p, add the connection
      (e_1 <-> $j_1...j_i$)
    add each alignment in p to result
  return result

Hint. An EM algorithm sketch

1. iteration = 0
2. for each word pair <$e_1...e_r$, $j_1...j_m$> in the corpus:
    2a. enumerate all legal alignments $a_1...a_n$
    2b. compute alignment probabilities $P(a_1)...P(a_n)$ as follows:
        - if (iteration == 0) then $P(a_i) = 1/n$ (for all I)
        - else,
            for i = 1 to n
              $P(a_i) = 1$
              for k = 1 to r
                let j...j be the Japanese sound sequence produced by English sound e_k in a_i
                $P(a_i) = P(a_i) * P(j...j \mid e_k)$
            / *normalize to yield $P(a_i)$ estimates* */*
            total = 0
            for i = 1 to n
              total = total + $P(a_i)$
            for i = 1 to n
              $P(a_i) = P(a_i)$ / total
            / *good place to print best alignment for this word pair* */*
    2c. collect sound-translation counts over all legal alignments as follows:
        for i = 1 to n
          for k = 1 to r
            let j...j be the Japanese sound sequence produced by English sound e_k in a_i
            count(j...j | e_k) += $P(a_i)$ */* *perhaps use a hash table?* */*
3. normalize sound-translation counts to yield P(j...j | e) estimates as follows:
    for each distinct English sound e
      total = 0
      for each distinct sequence j...j such that count(j...j | e) > 0 */* *perhaps map over hash table?* */*
        total += count(j...j | e)
      for each distinct sequence j...j such that count(j...j | e) > 0
        P(j...j | e) = count(j...j | e) / total
4. clear counts as follows:
    for each j…j and e such that count(j...j | e) > 0,
      count(j…j | e) = 0
5. iteration += 1
6. if (iteration < max-iterations), then goto step 2.