Homework #5: CMPT-825

Due in class on Oct 10, 2003

Anoop Sarkar – anoop@cs.sfu.ca

(1) (100pts) **Machine (Back) Transliteration**

All the files for this assignment are in `/cs/825/data/translit`

Languages have different sound inventories. When translating from one language to another, names, technical terms and even some common nouns are routinely transliterated. *Transliteration* means the replacement of a loan word with some approximate phonetic equivalents taken from the sound inventory of the language. These phonetic equivalents are then written in the script of the language.

For example, the word "computer" in English comes out sounding as "konpyutaa" in Japanese, which is written using the katakana syllabic script typically used for loan words. Here are some more examples of transliteration:

Angela Johnson
アンジラ・ジョンソン
(anjira jyonson)

New York Times
ニューヨーク・タイムズ
(nyuuyooku taimuzu)

ice cream
アイスクリーム
(aisukuriimu)

More information about this task and the particular approach we will be taking to solve this problem is described in the following paper:

- Machine Transliteration. Kevin Knight and Jonathan Graehl. *Computational Linguistics*. Vol 24, No. 4. December 1998. *available as the file* `transliterate.pdf`

Your task in this assignment is to back transliterate from Japanese into English. To avoid dealing with encoding issues with the Japanese script, we will assume that the input will be Japanese sound sequences rather than katakana characters. There is a simple mapping between katakana and these sound sequences shown in Figure 1 in the above paper.

The key to solve this problem is to represent the transliteration as a sequence of generative models (as defined using Bayes rule). You are given the parameters for each of these generative models. Once we have these models, we can represent each of them as a weighted finite-state transducer. We will do this using the AT&T `fsm` toolkit. The advantage is that we can use standard programs that are part of this toolkit in order to compose these finite state transducers into a single transducer which computes the conversion from Japanese sound sequences into English words.

The models we will use are as follows:

- $P(w)$ – provides a probability for English words, e.g.

  ```
  P(golf ball)
  ```

- $P(e \mid w)$ – provides a non-probabilistic mapping to English sound sequences given English words, e.g.

  `P(G AA L F B AO L | golf ball)`

- $P(j \mid e)$ – a map into Japanese sound sequences given English sound sequences, e.g.

  `P(g o r u h u b o o r u | G AA L F B AO L)`

- The input to the model is a particular Japanese sound sequence, e.g.

  `g o r u h u b o o r u`

We stop here for this assignment, but in general for this approach to be useful for text-text back transliteration we will have to build a generative model mapping Japanese sound sequences into the katakana script.
The training data corresponding to each stage above is in the following files:

- `engwords.dict` – a simple unigram model of English words
- `cmudict.0.6` – the CMU pronunciation dictionary for English words[1]
- `epron-jpron.map` – the mapping between English and Japanese sound sequences learned from data using the method defined in the Knight and Graehl paper.

These files are quite large, so in order to solve this assignment you should initially work with the following files which are smaller versions of the above files.

- `sample.engwords.dict`
- `sample.cmudict`
- `epron-jpron.map`

In order to do back transliteration from Japanese sound sequences into English words, you will have to convert the above files into the AT&T fsm toolkit format. You will need to create a text file with the finite state machine/transducer and an additional file with the mapping between symbols and numbers (the `.syms` file). *An important thing to keep in mind is that the same token must get the same number in the* `.syms` *file across all these FSMs.*
This conversion is not entirely trivial. For example, consider the fragment of `cmudict.0.6` given below:

```
ABBA  AE1 B AH0
ABBADO  AH0 B AA1 D OW0
ABBAS  AH0 B AA1 S
ABBASI  AA0 B AA1 S IY0
```

The numbers `0,1,2` are to be removed from this input (they represent stress markers which are not needed for this task). The first column is the English word and the remaining columns contain the pronunciation for that word.

[1]The English sounds (phonemes) used in this file are described in the file `phoneset.0.6`.

Once you write your code to convert the above format into the AT&T fsm toolkit, the transducer text file should look like this (call it `tiny.stxt`):

```
0 2 <eps> AE
2 3 <eps> B
3 4 <eps> AH
4 1 ABBA <eps>
0 6 <eps> AH
6 7 <eps> B
7 8 <eps> AA
8 9 <eps> D
9 10 <eps> OW
10 1 ABBADO <eps>
8 12 <eps> S
12 1 ABBAS <eps>
0 14 <eps> AA
14 15 <eps> B
15 16 <eps> AA
16 17 <eps> S
17 18 <eps> IY
18 1 ABBASI <eps>
1 0 <eps> <eps>
1 0 <eps> PAUSE
1
```

And the symbols file should look like this (called `tiny.syms`). The number `0` is reserved for the empty string, denoted here by `<eps>`:

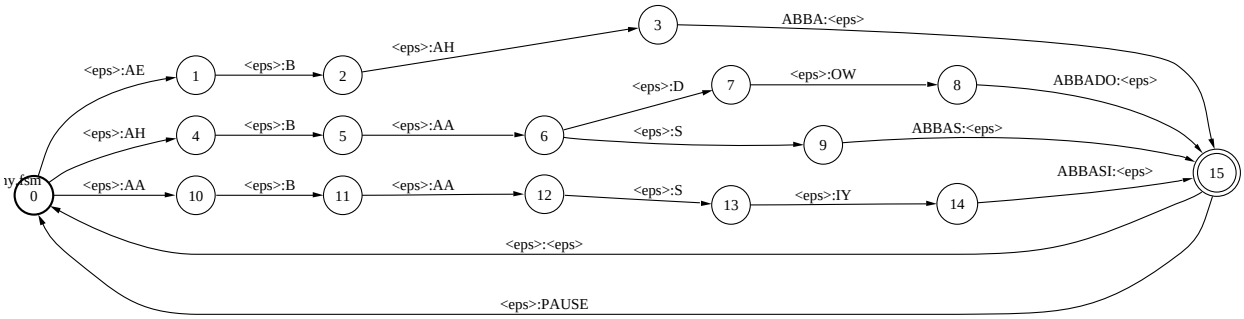
```
<eps> 0
PAUSE 1
AE 2
B 3
AH 4
ABBA 5
AA 6
D 7
OW 8
ABBADO 9
S 10
ABBAS 11
IY 12
ABBASI 13
```

Once you have these two files you can compile it using `fsmcompile`.

```
fsmcompile -t -i tiny.syms -o tiny.syms tiny.stxt > tiny.fsm
```

Graphically the transducer looks like this:[2]

[2]Using `fsmdraw -i tiny.syms -o tiny.syms tiny.fsm | dot -Tps > tiny.ps`



Once you have compiled all the individual finite state transducers for each model, you should use the command `fsmcompose` to combine them together into one big generative model which maps Japanese sounds into English words.

Once this is done you can then compose this combined fsm file with a particular input Japanese sound sequence and extract the best path from this using `fsmbestpath` combined with `fsmepsnormalize` and `fsmproject`. The output is an fsm which contains the most likely English word sequence.

The input Japanese sound sequence `anjira jyonson` represented as an fsm is given to you in the file `jpron-input.txt` and the syms file is in the file `jpron-syms.txt`.

Provide the entire sequence of AT&T fsm commands that you used in order to create the full generative model for back transliteration from Japanese sound sequences to English words.

Provide the commands used to find the best guess of the English words for the input Japanese sound sequence stored in the files `jpron-input.txt` and `jpron-syms.txt`.