Chapter 10

Extensions for Speech and Translation

10.1 Speech

We give only a brief overview of how to do automatic speech recognition using FSTs. For a more detailed treatment, see the survey by Mohri, Pereira, and Riley (2002).

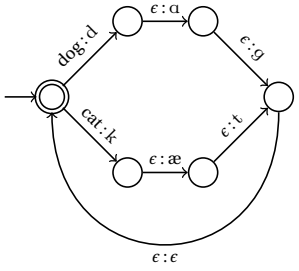
Given a sample of speech, we can convert it into a sequence of real-valued 39-dimensional vectors, called the *mel-frequency cepstral coefficients*:

1. Slice the signal into overlapping frames, typically 25 ms wide and starting every 10 ms.
2. Perform a Fourier transform on each frame, which is the amount of power at each frequency.
3. Compute how much power there is in each of several frequency bands arranged according to the *mel scale*, which is supposed to match human perception of pitch.
4. Take the log of each power, which matches human perception of loudness.
5. Take a *discrete cosine transform*, which removes correlations between the components.
6. Also compute the change of the components over time ($\Delta c_t = c_{t+2} - c_{t-2}$), and the change of the change ($\Delta^2 c_t = \Delta c_{t+1} - \Delta c_{t-1}$), for a total of 39 components.

We want to find the most likely sequence of words that this speech came from. To do this, we build a cascade of FSTs that maps from text to feature vectors:

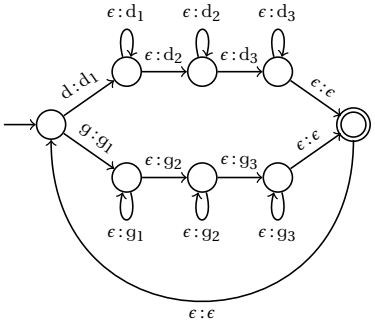
1. Generate words using a language model, typically an n-gram model.
2. Map words to their pronunciations, which are strings of *phones*.
3. Divide each phone into *subphones*, and stretch out each subphone to last for one or more frames.
4. For each frame, map the subphone to a feature vector.

We've seen FSAs for n-gram models already. To maps from words to their pronunciations (shown just for two words), we can use an FST like this:

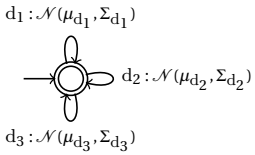


Hand-built dictionaries are available (e.g., the CMU pronunciation dictionary), or a grapheme-to-phoneme model could be used or even learned automatically.

The next stage divides each phone into *subphones* and decides the duration of each subphone. Typically, each phone is divided into three subphones; for example, phone d would be divided into d_1, d_2, and d_3. The duration of each subphone is one or more time slices. So the FST looks like this (shown just for two phones):



Finally, we map from the sequence of subphone symbols to the observed sequence of feature vectors. The problem is that the vectors have real-valued components, but we only know how to define FSTs with a finite output alphabet. So we have to generalize our notation. If we write $a : P(B \mid a)$ on an arc, that means that on input a, the FST outputs a symbol B drawn from the distribution $P(B \mid a)$, and this distribution can be defined on an arbitrary space. In effect, this is equivalent to an infinite number of arcs: for each b, an arc $a : b$ with weight $P(b \mid a)$. Thus (shown just for one phone):



Typically, the last two FSTs (phone to subphone and subphone to feature vectors) are composed and trained together using EM (Forward-Backward).

10.2 Word Alignment

10.2.1 Problem

A *parallel text* is a corpus of text that expresses the same meaning in two (or more) different languages. Usually we assume that a parallel text is already *sentence-aligned*, that is, it consists of *sentence pairs*, each of which expresses the same meaning in two languages. Conventionally, following Brown et al. (1993), the two languages are referred to as English and French even when other languages are possible. Here, we use English and Spanish.

Here is an example parallel text:

1. garcia and associates
   garcia y asociados
2. his associates are not strong
   sus asociados no son fuertes
3. the groups do not sell zenzanine
   los grupos no venden zanzanina

The *word alignment* problem is to figure out which Spanish words correspond to which English words. This would be the correct word alignment for our example:

1. garcia and associates
   garcia y asociados
2. his associates are not strong
   sus asociados no son fuertes
3. the groups do not sell zenzanine
   los grupos no venden zanzanina

(Aligning *do* to *venden* in 3 would also be correct.)

More formally: let

- $\mathbf{f} = f_1 \cdots f_m$ range over Spanish sentences
- $\mathbf{e} = e_1 \cdots e_\ell$ range over English sentences
- $\mathbf{a} = (a_1, \ldots, a_m)$ range over possible many-to-one alignments, where $a_j = i$ means that Spanish word j is aligned to English word i, and $a_j = \text{NULL}$ means that Spanish word j is unaligned. Thus the alignment for sentence (2) above is $(1, 2, 4, 3, 5, 6)$.

We are given a sequence of $(\mathbf{f}, \mathbf{e})$ pairs. We are going to define a model of $P(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$ and our job is to estimate the parameters of the model to maximize the likelihood $P(\mathbf{f} \mid \mathbf{e})$.

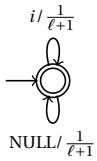
10.2.2 Model 1

IBM Model 1 (Brown et al., 1993) is the first in a series of five seminal models for statistical word alignment. The basic generative story goes like this:

1. Choose m with uniform probability $\epsilon = \frac{1}{M}$, where M is the maximum length of any Spanish sentence in the corpus.
2. Generate an alignment $a_1, \ldots, a_m$, again with uniform probability.
3. Generate Spanish words $f_1, \ldots, f_m$, each with probability $t(f_j \mid e_{a_j})$ or $t(f_j \mid \text{NULL})$.

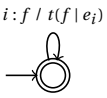
There isn't a one-size-fits-all finite-state machine that computes Model 1 for any sentence pair. Instead, for each **e**, we can make a FSA that generates French sentences **f** according to the Model 1 probability $P(\mathbf{f} \mid \mathbf{e})$. It is a cascade of steps 2 and 3 above.

The alignment-generation model can be written as a very simple FSA:



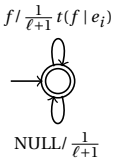
The arc labeled i is actually many arcs, one for each $1 \le i \le \ell$. Notice that because m is determined beforehand, this FSA doesn't need a stop probability.

The Spanish-word-generation model is also very simple:



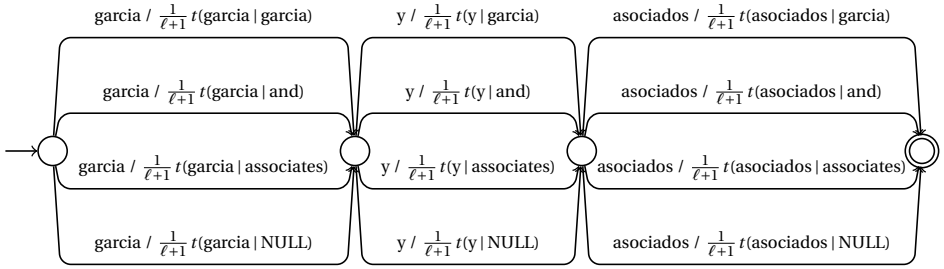
The arc labeled $i : f$ is actually many arcs, one for every Spanish word f and every English position $1 \le i \le \ell$.

Composing the two, we get IBM Model 1.



The arc labeled f is actually many arcs, one for every Spanish word f and every English position $1 \le i \le \ell$. This automaton can generate any Spanish string **f** with any alignment to **e**, and the probability of the path is $P(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$. Note that although we can generate any Spanish string, the English string **e** remains fixed.

On intersecting this FSA with **f**, we get:

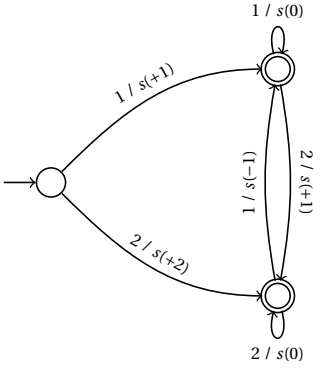


Then the $t(f \mid e)$ can be estimated using Expectation-Maximization.

1. Initialize $t(\cdot \mid e)$ to uniform.
2. E-step:
   (a) Use the Forward-Backward algorithm to calculate a fractional count for each arc.
   (b) For each arc with weight $\frac{1}{\ell+1}\,t(f \mid e)$ and fractional count p, let $c(f,e) \leftarrow c(f,e) + p$.
3. M-step: let $t(f \mid e) \leftarrow \frac{c(f,e)}{\sum_f c(f,e)}$.
4. Go to 2.

10.2.3 Hidden Markov Model

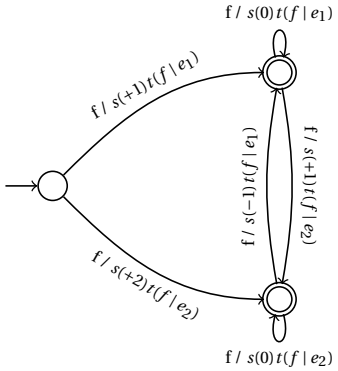
Model 1 doesn't care at all about word order. But we would like to capture the fact that if a Spanish word is translated from an English word, the next Spanish word is probably translated from the next English word. So we need some kind of dependence between the a_j. One easy and efficient way to do this is to make an alignment dependent on the previous alignment, i.e., a_j depends on a_{j-1} (Vogel, Ney, and Tillmann, 1996). Just as a bigram model could be represented as an FSA, so can this model. Here we show the FSA for $\ell = 2$.



The distribution $s(\Delta i)$ gives the probability that, if a Spanish word is translated from English word e_i, the next Spanish word is translated from English word $e_{i+\Delta i}$. We are treating the start of the sentence as position 0. So, if $\mathbf{a} = (1,2,4,3,5,6)$, its probability is $s(+1)s(+1)s(+2)s(-1)s(+2)s(+1)$. We expect the distribution to peak at +1 and decrease for larger $|\Delta i|$.

The model is deficient since it assigns a nonzero probability to alignments that fall off the edge of the sentence. We can fix this by renormalizing, but we haven't bothered to do so for simplicity's sake. Also note that there are no NULL alignments. These can be added in but were omitted in the original version (Vogel, Ney, and Tillmann, 1996) which we follow.

We compose this FSA with the same Spanish-word-generation model that we used before (again, assuming $\ell = 2$):



Then we compose with **f** (we don't show the result here because it would be too complicated). Then we have to estimate both the $t(f \mid e)$ and the $s(\Delta i)$. The algorithm goes like this:

1. Initialize $t(\cdot \mid e)$ and $s(\cdot)$ to uniform.
2. E-step:
   (a) Use the Forward-Backward algorithm to calculate a fractional count for each arc.
   (b) For each arc with weight $s(\Delta i)\, t(f \mid e)$ and fractional count p, let
   $$c_s(\Delta i) \leftarrow c_s(\Delta i) + p$$
   $$c_t(f, e) \leftarrow c_t(f, e) + p$$
3. M-step: let
   $$s(\Delta i) \leftarrow \frac{c_s(\Delta i)}{\sum_{\Delta i} c_s(\Delta i)}$$
   $$t(f \mid e) \leftarrow \frac{c_t(f, e)}{\sum_f c_t(f, e)}$$
4. Go to 2.

References

Brown, Peter F. et al. (1993). "The Mathematics of Statistical Machine Translation: Parameter Estimation". In: *Computational Linguistics* 19, pp. 263–311.

Mohri, Mehryar, Fernando Pereira, and Michael Riley (2002). "Weighted Finite-State Transducers in Speech Recognition". In: *Computer Speech and Language* 16.1, pp. 68–88.

Vogel, Stephan, Hermann Ney, and Christoph Tillmann (1996). "HMM-Based Word Alignment in Statistical Translation". In: *Proc. COLING*, pp. 836–841.