Chapter 8

Word Alignment

8.1 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 $P(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$.

All the IBM models (Brown et al., 1993) and their offspring can be broken into three parts:

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = \frac{1}{M} \times P(\mathbf{a} \mid \mathbf{e}, m) \times \prod_{j=1}^{m} \begin{cases} t(f_j \mid \text{NULL}) & \text{if } a_j = \text{NULL} \\ t(f_j \mid e_{a_j}) & \text{otherwise} \end{cases}$$

The three factors correspond to three steps:

1. Choose m, the length of the Spanish sentence, with uniform probability $\frac{1}{M}$, where M is some maximum length. (It doesn't matter what M is as long as it's bigger than any actual Spanish sentence in the data).
2. Generate an alignment $a_1, \ldots, a_m$. You can think of this part as a sequence labeling problem, where we are trying to label the *Spanish* words, and the labels are drawn from $\{1, \ldots, \ell, \text{NULL}\}$, where ℓ is the length of the English sentence.
3. Generate Spanish words $f_1, \ldots, f_m$, each with probability $t(f_j \mid e_{a_j})$ or $t(f_j \mid \text{NULL})$. (Here, $t(f \mid e)$ is a conditional probability distribution, but it's conventional to call it t to avoid overloading p.)

The models differ only in step 2.

In order to train these models, we want to maximize the log-likelihood,

$$\log P(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}).$$

In general, the summation over all possible alignments $\mathbf{a}$ is intractable. This is the main technical challenge.

8.2 IBM Model 1

IBM Model 1 is the simplest of the five IBM models. In this model, we have

$$P(\mathbf{a} \mid \mathbf{e}, m) = \prod_{j=1}^{m} \frac{1}{\ell + 1}. \tag{8.1}$$

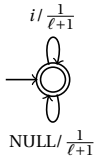
This submodel has no parameters. It just says that for each Spanish word, we choose a corresponding English word (or NULL) with uniform probability.

As noted above, to train this model, we need to compute the log-likelihood, which involves a summation over all possible alignments $\mathbf{a}$. We can do this using the tools we've already learned, if we can

write the model as a weighted finite automaton. So, given **e**, we can make a WFA that generates Spanish sentences **f**.

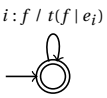
Step 1 is not important and we can safely ignore it.

Step 2, the alignment-generation model, can be written as a very simple WFA:



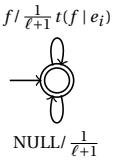
The arc labeled i is actually many arcs, one for each $1 \le i \le \ell$. This automaton just emits a sequence of English word positions with uniform probability. Notice that because m is determined beforehand, this automaton doesn't need a stop probability.

Step 3, the Spanish-word-generation model, is also very simple:



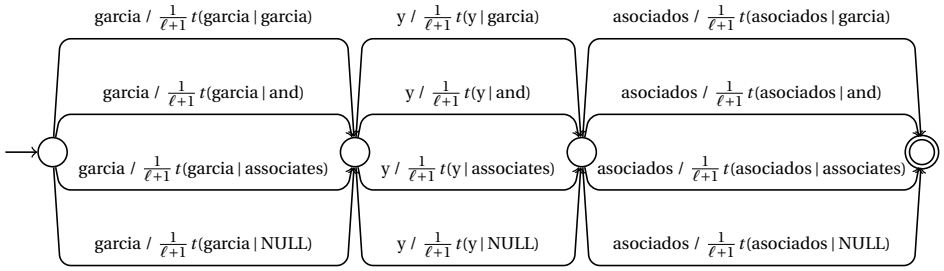
The arc labeled $i : f$ is actually many arcs, one for every Spanish word f and every English position $1 \le i \le \ell$.

Composing the two, we get IBM Model 1, which is still pretty simple:



The arc labeled f is actually many arcs, one for every Spanish word f and every English position $1 \le i \le \ell$. Every path through this automaton corresponds to an alignment **a** and a Spanish string **f**, 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. If we were to sum the probabilities of all the paths, they would sum to one.

Now we intersect this WFA with **f** = "garcia y asociados" because we're only interested in the paths that generate **f**, so we can compute $P(\mathbf{f} \mid \mathbf{e})$. That gives:



Then, $P(\mathbf{f} \mid \mathbf{e})$ is the sum of all the probabilities of this automaton. We can compute this using the Forward algorithm. Specialized for Model 1, it looks like this:

forward[0] $\leftarrow 1$

for $j \leftarrow 1, \ldots m$ **do**
  forward$[j] \leftarrow 0$
  for $i \leftarrow 1, \ldots \ell$ **do**
    forward$[j] \leftarrow$ forward$[j]$ + forward$[j-1] \times \frac{1}{\ell+1} t(f_j \mid e_i)$
  end for
  forward$[j] \leftarrow$ forward$[j]$ + forward$[j-1] \times \frac{1}{\ell+1} t(f_j \mid \text{NULL})$
end for

Then $P(\mathbf{f} \mid \mathbf{e}) = \text{forward}[m]$.

Now, to train the model by stochastic gradient ascent, we express the probabilities $t(f \mid e)$ in terms of unconstrained variables $\lambda(f \mid e)$:

$$t(f \mid e) = \frac{\exp \lambda(f \mid e)}{\sum_{f'} \exp \lambda(f' \mid e)} \tag{8.2}$$

And the pseudocode for the optimization is:

randomly initialize $\lambda(f \mid e)$ for all f, e
repeat
  $LL \leftarrow 0$ ▷ log-likelihood $\log L$
  for training examples $\mathbf{f}, \mathbf{e}$ **do**
    compute $\log P(\mathbf{f} \mid \mathbf{e})$
    $LL \leftarrow LL + \log P(\mathbf{f} \mid \mathbf{e})$
    compute $\frac{\partial \log P(\mathbf{f} \mid \mathbf{e})}{\partial \lambda(f \mid e)}$ at the current point
    $\lambda(f \mid e) \leftarrow \lambda(f \mid e) + \eta \cdot \frac{\partial \log P(\mathbf{f} \mid \mathbf{e})}{\partial \lambda(f \mid e)}$
  end for
  print LL ▷ should increase
until done

In the rest of the chapter, we give just a brief overview of some more advanced alignment models.

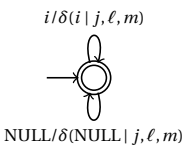
8.3 IBM Model 2

Model 1 doesn't care at all about word order. 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. Model 2 is a small step in that direction. It replaces the uniform alignment model with:

$$P(\mathbf{a} \mid \mathbf{e}, m) = \prod_{j=1}^{m} \delta(a_j \mid j, \ell, m) \tag{8.3}$$

where $\delta(a_j \mid j, \ell, m)$ is a probability distribution.

As a WFA, the alignment-generation model looks like:

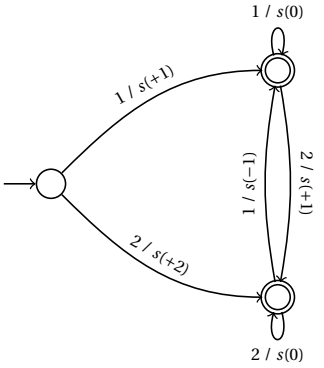


The arc labeled i is actually many arcs, one for each $1 \le i \le \ell$.

8.4 Hidden Markov Model

Even though Model 2 captures something about word order, it still doesn't know anything about the relationships between adjacent alignments. It is closer in spirit to the bag-of-words models we started the semester with than the sequence models we've been looking at more recently.

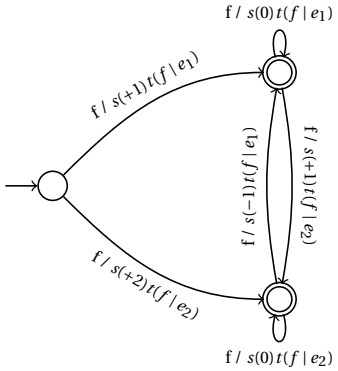
One easy and efficient way to improve the model is to make an alignment dependent on the previous alignment, i.e., a_j depends on a_{j-1} (Vogel, Ney, and Tillmann, 1996). Just as a bigram model could be represented as a WFA, so can this model. Here we show the automaton for $\ell = 2$.



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 here.

We compose this WFA 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 $s(\Delta i)$.

Bibliography

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

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