Translation with Finite-State Devices

Kevin Knight and Yaser Al-Onaizan

Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
knight@isi.edu, yaser@isi.edu

Abstract. Statistical models have recently been applied to machine translation with interesting results. Algorithms for processing these models have not received wide circulation, however. By contrast, general finite-state transduction algorithms have been applied in a variety of tasks. This paper gives a finite-state reconstruction of statistical translation and demonstrates the use of standard tools to compute statistically likely translations. Ours is the first translation algorithm for "fertility/permutation" statistical models to be described in replicable detail.

1 Introduction

Statistical machine translation (SMT) is a relatively recent approach to the longstanding problem of translating human languages by computer. The designer of an SMT system constructs a general model of how the translation process works, then lets the system acquire specific rules automatically from bilingual and monolingual text corpora (see Figure 1 and 2). These rules are usually coarse and probabilistic—for example, in a certain corpus, *bat* translates as *palo* 71% of the time, and as *murciélago* 29%. The most-established SMT system is based on word-for-word substitution (Brown *et al.*, 1993), although some experimental SMT systems employ syntactic and semantic processing (Wu, 1997; Alshawi *et al.*, 1997; Su *et al.*, 1995). An advantage of the SMT approach is that designers can improve translation accuracy by modestly changing the underlying model rather than overhauling large handcrafted resources.

Unfortunately, SMT has had little impact on machine translation in practice. One reason is that SMT accuracy is not significantly better than that obtained by handcrafting. Another reason is that few people understand SMT techniques like those presented in (Brown *et al.*, 1993), because of their highly mathematical rather than linguistic nature. These mathematical difficulties have slowed duplication and improvement of SMT results.

By contrast, finite-state methods are having a broad effect on various aspects of human language processing (Roche and Schabes, 1997). Techniques for building and composing automata are widely understood, and software toolkits are becoming available to researchers. Finite-state machines offer advantages such as speed, precision, and smooth integration with speech and character recognition.

```
Garcia and associates .
Garcia y asociados .

his associates are not strong .
sus asociados no son fuertes .

the company has three groups .
la empresa tiene tres grupos .

its groups are in Europe .
sus grupos estan en Europa .

its clients are angry .
sus clientes estan enfadados .

the associates are also angry .
los asociados tambien estan enfadados .

the clients and the associates are enemies .
los clientes y los asociados son enemigos .

Carlos Garcia has three associates .
Carlos Garcia tiene tres asociados .

the modern groups sell strong pharmaceuticals .
los grupos modernos venden medicinas fuertes .

Garcia has a company also .
Garcia tambien tiene una empresa .

the groups do not sell zanzanine .
los grupos no venden zanzanina .

the small groups are not modern .
los grupos pequenos no son modernos .
```

Fig. 1. Sample bilingual text corpus used in training SMT systems (adapted from Knight, 1997)

The aim of this paper is to give a finite-state reconstruction of SMT, with three goals in mind. One is to present a new implementation for SMT in which standard algorithms are used to compute probabilities. Another is to present the first translation algorithm for "fertility/permutation" statistical models to be described in replicable detail. Finally, we aim to make SMT techniques more accessible to the wider community.

```
Carlos has strong clients .
its small groups have strong enemies .
Carlos is angry with his enemies in Europe who also sell
      pharmaceuticals .
the pharmaceuticals are with Carlos .
Garcia and associates .
Carlos Garcia has three associates .
his associates are not strong .
Garcia has a company also .
its clients are angry .
the associates are also angry .
the clients and the associates are enemies .
the company has three groups .
its groups are in Europe .
the modern groups sell strong pharmaceuticals .
the groups do not sell zanzanine .
```

Fig. 2. Sample monolingual text corpus used in training SMT systems (adapted from Knight, 1997).

2 Word-for-Word SMT

SMT views any string of words e as a potential translation of any string f.[1] We would like to set up a probability distribution $\mathrm{P}(e|f)$ over all pairs of strings, so that given f, we can output the e which maximizes $\mathrm{P}(e|f)$. We break this problem down into two simpler problems by using Bayes' Rule:

$$\mathrm{P}(e \mid f) \sim \mathrm{P}(e) \cdot \mathrm{P}(f \mid e)$$

The $\mathrm{P}(e)$ factor helps our e output to be natural and grammatical, while the $\mathrm{P}(f|e)$ factor ensures that e is normally interpreted as f, and not some other thing.

It is interesting that reasonable translations can come from highly flawed $\mathrm{P}(e)$ and $\mathrm{P}(f|e)$ estimates. For example, Figure 3 shows several English translations of some French sentence. Translations that pass $\mathrm{P}(e)$ are marked with **x** in the first column; likewise for translations that pass $\mathrm{P}(f|e)$.

Both models are flawed. $\mathrm{P}(e)$ assigns too much probability to *TV appeared on Jon*, while $\mathrm{P}(f|e)$ assigns too much to *Jon appeared in TV*. However, the translation that they both agree on is very reasonable. One goal of SMT is to find probability estimates that are "good enough" to yield acceptable translations.

A common model for $\mathrm{P}(e)$ is the word-ngram model, which assigns values to strings based on their sub-sequence frequencies. We turn to models for $\mathrm{P}(f|e)$ in the next section.

[1] We follow the (Brown *et al.*, 1993) terminology of denoting natural language strings with letters e and f, even if the languages are not English and French. We also refer to e as the source sentence and f as the target sentence, even though the top-level goal will be to translate from f to e.

	P(e)	P(f|e)
Jon appeared in TV.		x
Appeared on Jon TV.		
In Jon appeared TV.		x
Jon is happy today.	x	
Jon appeared on TV.	x	x
TV appeared on Jon.	x	
TV in Jon appeared.		
Jon was not happy.	x	

Fig. 3. English translations of a French sentence scored with models $P(e)$ and $P(f|e)$

Once the general models are in place, we need a training algorithm to fix their parameters, and a "decoding" algorithm for finding the e that maximizes the value of the formula above. We will discuss decoding in our finite-state reconstruction.

3 Translation Models

This section summarizes translation Models 1-3 as presented in (Brown *et al.*, 1993). Each model specifies how to compute a value for any $P(f|e)$ by consulting tables of probabilities. In all three models, a source sentence may produce the same target sentence in several different ways—these ways are covered by different *alignments*. An alignment is a set of connections between source words and target words, normally such that each target word is connected to exactly one source word. We assume that an invisible NULL word exists at the beginning of each source sentence to pick up any "spurious" target words. Here is an alignment between *an old house* and *una casa vieja*:

```
NULL  an  old  house
 |     |     \ /
       |      X
       |     / \
      una casa vieja
```

And here is another, less probable alignment:

```
NULL  an  old  house
 |    |    |      |
 |       +-+-+
 |       |   |
una   casa  vieja
```

Supposing that e contains l words and f contains m words, there are $(l+1)^m$ conceivable alignments. We refer to an alignment as a, and can write it as a series of values a_j $(j = 1, 2, \ldots m)$, each of which records the source language position that target word j connects to. The job of computing $\mathrm{P}(f|e)$ thus becomes:

$$P(f \mid e) = \sum_a P(f, a \mid e)$$

where a ranges over all the conceivable alignments for the pair of sentences. Each model defines the probability of an alignment as a function of the connections it contains, and each does so in a different way.

3.1 Model 1

The first model is based on the following hypothetical translation process: given a source sentence of length l, pick a desired target sentence length m; then, for each target position $j = 1 \ldots m$ choose a connection to source position a_j; and for each target position $j = 1 \ldots m$ choose a target word f_j.

Connections are chosen from the uniform distribution, i.e., a_j may take on any value from 0 to l with equal probability. Word choices are made according to a probabilistic table

$\mathrm{t}(f|e)$ = probability that word e gets translated as f

We can write

$$\mathrm{P}(f, a \mid e) = c \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

where alignment a is a series of values a_j $(j = 1 \ldots m)$, and c relates to the positional connections and to choice of m itself.

To get a formula for computing $\mathrm{P}(f|e)$, we sum over all possible alignments a, as shown above.

Notice that in this hypothetical process there is no notion of consuming the source sentence word by word and producing the target sentence. Instead, all source words must remain available for consultation. Also notice the process chooses a number m which subsequent steps must remember and adhere to. These properties make Model 1 unattractive for finite-state modeling.

3.2 Model 2

This model is the same as Model 1 except that the value of an a_j is not chosen uniformly. It depends on j, as specified in another probabilistic table:

$a(i \mid j, l, m)$ = alignment probability of position j

This allows the model to express a preference that words at the beginning of a target sentence tend to get connected to words at the beginning (or end) of a source sentence. Here

$$\mathrm{P}(f, a \mid e) = c \prod_{j=1}^{m} t(f_j \mid e_{a_j}) \prod_{j=1}^{m} a(a_j \mid j, l, m)$$

3.3 Model 3

Model 3 is based on a very different hypothetical translation process. For each e_i, we choose a *fertility* ϕ_i that dictates how many target words will connect to it. For each $i = 0, 1, \ldots l$ and each $k = 1, 2, \ldots \phi_i$, we then choose a target word τ_{ik}. Finally, we permute all these words.

Fertilities for $e_1 \ldots e_l$ are prescribed by the probabilistic table

$n(\phi \mid e)$ = fertility of word e

The fertility of the NULL word (e_0) depends on the length of the target sentence, because longer sentences have more spurious words. Model 3 introduces a single probability

p_1 = NULL word parameter

and its opposite $p_0 = 1 - p_1$. A fertility of ϕ_0 is selected with probability

$$\binom{m'}{\phi_0} p_0^{m'-\phi_0} p_1^{\phi_0}$$

$m' = \sum_{i=1}^{l} \phi_i$ is used as an approximate target-sentence length, because m itself is unknown until we choose ϕ_0. Actual target words τ_{ik} are chosen from table $t(f|e)$ as in Model 1.

The final permutation assigns a value π_{ik} (from 1 to m) to each target word. Model 3 reverses the position-mapping probabilities from Model 2, so that π_{ik} ($i > 0$) are assigned according to the table

$d(\pi_{ik} \mid i, l, m)$ = distortion of position i

Words generated from NULL ($i = 0$) are free to roam unbound by the d-table. Of the ϕ_0 spots left vacant in $\pi_i k$, there are $\phi_0!$ ways to arrange the NULL-generated words. A particular selection contributes an additional $1/\phi_0!$ factor to the d probabilities.

The target sentence has $f_{\pi_{ik}} = \tau_{ik}$ for each $i = 0, 1, \ldots l$ and each $k = 1, 2, \ldots \phi_i$.

For any source sentence, a choice of fertilities, translations, and permutation pick out both a target sentence and an alignment. For example, if e = *an old house* and we choose $\phi_0 = 0$, $\phi_1 = 1$, $\phi_2 = 1$, $\phi_3 = 1$, τ_{11} = *una*, τ_{21} = *vieja*, τ_{31} = *casa*, $\pi_{11} = 1$, $\pi_{21} = 3$, $\pi_{31} = 2$, then we obtain the first of the two alignments shown in Section 3.

Several sets of choices may correspond to the same alignment. For example, the alignment

```
    NULL  seatbelt
              |
   +---------+----------+
   |         |          |
cinturon    de    seguridad
```

seems to require

$$\tau_{11} = \textit{cinturon}, \tau_{12} = \textit{de}, \tau_{13} = \textit{seguridad}$$
$$\pi_{11} = 1, \pi_{12} = 2, \pi_{13} = 3$$

but

$$\tau_{11} = \textit{de}, \tau_{12} = \textit{cinturon}, \tau_{13} = \textit{seguridad}$$
$$\pi_{11} = 2, \pi_{12} = 1, \pi_{13} = 3$$

produces the same thing.

(Brown *et al.*, 1993) note that the permutation process can generate non-strings—if every π_{ik} is chosen to be 5, then all target words will be piled up in position 5. Model 3 is therefore said to be *deficient.*

All this leads to:

$$\begin{aligned} \mathrm{P}(f, a \mid e) = & \binom{m - \phi_0}{\phi_0} {p_0}^{m-2\phi_0} {p_1}^{\phi_0} \cdot \\ & \prod_{i=1}^{l} n(\phi_i \mid e_i) \prod_{i=0}^{l} \phi_i! \cdot \\ & \prod_{j=1}^{m} t(f_j \mid e_{a_j}) \cdot \\ & \tfrac{1}{\phi_0!} \cdot \prod_{j: a_j \neq 0} d(j \mid a_j, l, m) \end{aligned}$$

This formula, somewhat daunting at first, is explained by the finite-state reconstruction of Section 6. There we achieve the same values for $\mathrm{P}(f, a|e)$ without carrying out any combinations or factorials, and we compute $\mathrm{P}(f|e)$ without explicit reference to alignments at all.

4 Alternative Models

We will look at two alternatives to Model 3 not documented in (Brown *et al.*, 1993). One replaces the $d(\pi_{ik}|i, l, m)$ table by a simpler $d(\pi_{ik}|i)$ table. The other disposes of distortion probabilities altogether. Each of the $m!$ possible permutations is deemed equally likely (probability $= 1/m!$), and

$$\begin{aligned} \mathrm{P}(f, a \mid e) = & \binom{m - \phi_0}{\phi_0} {p_0}^{m-2\phi_0} {p_1}^{\phi_0} \cdot \\ & \prod_{i=1}^{l} n(\phi_i \mid e_i) \prod_{i=0}^{l} \phi_i! \cdot \\ & \prod_{j=1}^{m} t(f_j \mid e_{a_j}) \cdot \\ & \tfrac{1}{m!} \end{aligned}$$

This last version of Model 3 is not deficient—it does not allow target words to be stacked on top of one another. Of course, it does not capture word-order effects in translation, leaving source language word order up to the language model $P(e)$.

5 Finite-State Devices

A weighted finite-state transducer is one way of implementing a probability distribution over pairs of strings. It is a state-transition diagram whose arcs have labels of the form $x{:}y/w$, where x is an input word, y is an output word, and w is a numerical weight. Either x or y may be the empty token ϵ.

To get $\mathrm{P}(f|e)$, we find a path whose input-word sequence matches e and whose output-word sequence matches f, then multiply the weights of the arcs along that path. Actually, we must find *all* such matching paths and sum their contributions—there may be more than one way to produce f from e, just as there is more than one way to throw a seven with two dice. To ensure compliance with the laws of probability, we require that all arcs leaving state s with input word x have w's that sum to one.

Transducers can be composed algorithmically (Pereira and Riley, 1997). If one transducer implements $\mathrm{P}(x|y)$ and another implements $\mathrm{P}(y|z)$, then we can compute a third transducer implementing $\mathrm{P}(x|z)$. This allows a cascade of small transducers to simulate a big one.

If we supply an input string to a transducer, we get a network of all possible outputs, with arc-probabilities. Likewise, if we supply an output string, we get all possible inputs. More generally, we can supply a network of possible outputs and get a network of possible inputs.

Finding a most-likely input or output sequence (also known as "decoding") is a kind of shortest-path graph problem.

6 Finite-State SMT

We will now effect the Model 3 translation process using finite-state devices. Rather than constructing one giant transducer, we will construct a cascade of simple transducers that ultimately leads us from a source sentence to a target sentence.

Figure 4 shows the plan. There are four transducers in the translation model. If we insert *Mary did not slap the green witch* at the top of the cascade, many Spanish sentences could come out the bottom, each with probability $\mathrm{P}(f|e)$. One is shown here. Likewise, if we insert *Mary no daba bofetada a la bruja verde* at the bottom, many English sentences could come out the top, each with its own $\mathrm{P}(f|e)$.

Model 3 does not generate very good target sentences, even when the permutation process is moderated by the distortion table. So in practice, we will use the cascade only in these ways:

$\boxed{\text{Transducer Zero}}$

$\Downarrow$

Mary did not slap the green witch

$\Uparrow$

$\boxed{\text{Transducer One}}$

$\Downarrow$

Mary not slap slap slap the green witch

$\Uparrow$

$\boxed{\text{Transducer Two}}$

$\Downarrow$

Mary not slap slap slap NULL the green witch

$\Uparrow$

$\boxed{\text{Transducer Three}}$

$\Downarrow$

Mary no daba una botefada a la verde bruja

$\Uparrow$

$\boxed{\text{Transducer Four}}$

$\Downarrow$

Mary no daba una botefada a la bruja verde

Fig. 4. SMT as a cascade of finite-state transducers.

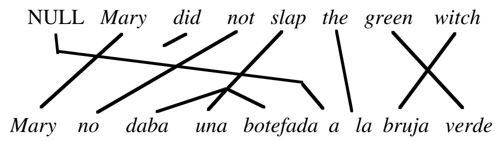


Fig. 5. One alignment for a pair of sentences.

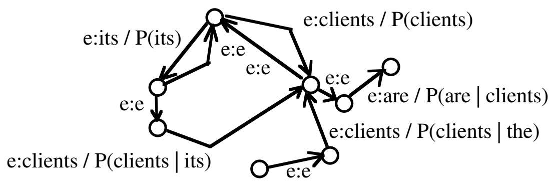
- For a given pair (e, f), estimate $\mathrm{P}(f|e)$.
- For a given target sentence f, find the most likely source sentence ê:

$$\hat{\mathrm{e}} = \underset{e}{\mathrm{argmax}}\, \mathrm{P}(e) \cdot \mathrm{P}(f \mid e)$$

For the latter task, we add a source language model to the top of the cascade.

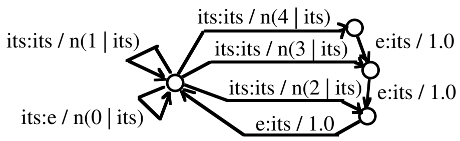
6.1 Transducer Zero

This transducer—actually an acceptor—assigns a probability to any source sentence. Although many acceptors are possible, we will assume a word-trigram model with *deleted interpolation* smoothing (Jelinek and Mercer, 1980). Here is a piece of this acceptor:



6.2 Transducer One

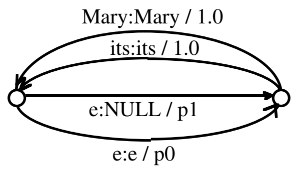
This transducer determines fertilities for words on its input. If the fertility of an input word is zero, the word is dropped in the output (e.g., *did*). If the fertility is one, it is copied. Otherwise, the word is duplicated, triplicated, etc. Here is a piece of this transducer:



We might say this transducer converts English to *Enggglis*. The connection between English and Enggglis sentences is usually clear, although in an occasional pair like *I had had it* and *I had had had it*, it is not obvious what fertilities were employed.

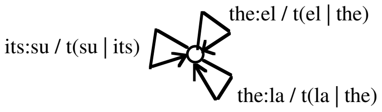
6.3 Transducer Two

This transducer sprinkles NULL words throughout an Enggglis string. More precisely, before copying each input word to the output, it emits NULL with probability p_1. Here is a piece:



6.4 Transducer Three

Here, we simply substitute target words for source words, one-for-one, according to $t(f|e)$:

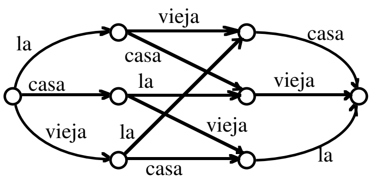


We can call the output *Spanglish*, because it is Spanish written with English word order.

6.5 Transducer Four

The final task is to permute a Spanglish sentence into a Spanish one. Here we will assume that all permutations are equally likely, as in Section 4. Unfortunately, no practical finite-state transducer can do this work. But because we are at the end of the road, we can do this with a non-finite-state algorithm. Given a particular Spanish sentence, we build a finite-state acceptor that includes all permutations of it. Each permutation is a potential Spanglish antecendent. Of particular interest are the ones for which simple word substitution yields good English.

Here is a permutation acceptor for *la casa vieja*:



Permutation acceptors have 2^m states.

6.6 Computation of P(f, a | e)

Next, we determine the probability that a particular target sentence will be generated from a particular source sentence. We first confine ourselves to transductions that correspond to a single alignment a. Recall that the same alignment may be produced from different choices of word translations (τ) and permutations (π).

If we take a to be the alignment shown in Figure 5, then there are 48 ways to produce *Mary no daba una bofetada a la bruja verde* from *Mary did not slap the green witch*. To see this, first consider that *slap* could translate to *bofetada daba no* in Spanish, and get permuted back to *no daba bofetada*. There are 6 (3!) such ways this could happen. Also consider that NULL could appear in any of eight positions, produce the Spanish word a, then get permuted back into the fifth position in the Spanish output. Each of the 48 sequences has the same probability, so that:

$$\mathrm{P}(f, a \mid e) = 48 \cdot \prod_{i=1}^{7} n(\phi_i \mid e_i) \cdot {p_0}^7 {p_1}^1 \cdot \prod_{j=1}^{9} t(f_j \mid e_{a_j}) \cdot \frac{1}{9!}$$

This is the same value we get from the $\mathrm{P}(f, a|e)$ in Section 4:

$$P(f, a \mid e) = \binom{9-1}{1} {p_0}^7 {p_1}^1 \cdot \prod_{i=1}^{7} n(\phi_i \mid e_i) \prod_{i=0}^{7} \phi_i! \cdot \prod_{j=1}^{9} t(f_j \mid e_{a_j}) \cdot \frac{1}{9!}$$

because $\phi_4 = 3$. Tracing the finite-state composition gives an intuitive picture of the factorials and combinations in the $\mathrm{P}(f, a|e)$ formula.

6.7 Computation of P(f | e)

To compute $\mathrm{P}(f|e)$ using the finite-state cascade, we create two acceptors, one for e (single-path), and one for f (permutational), add them to the beginning and end of the cascade (omitting Transducer Zero), then compose all six automata. The resulting transducer will contain a path for each way of producing f from e. To get $\mathrm{P}(f|e)$, we sum the weights of all these paths. This computation makes no reference to alignments, but computes the same value as summing $\mathrm{P}(f, a|e)$ over all possible alignments a.

7 Translation Algorithm

In this section, we show how sample translations are produced by the finite-state cascade. We have trained a translation Model 3 on the bilingual corpus shown in Figure 1, and a trigram language model on the monolingual corpus in Figure 2. We store the training results as finite-state automata.[2]

Computing the most-likely source language translation for f is similar to computing $\mathrm{P}(f|e)$. We don't know e, but we can constrain it using a source language model. We again compose all six automata, but instead of summing the path weights, we extract the k highest-probability paths. This gives us a k-best list of possible translations.

One of our aims in giving this Model 3 translation algorithm is to fill a gap in the translation algorithm ("decoding") literature. Unfortunately, (Brown *et al.*, 1993) never published a replicable fertility/distortion translation algorithm, and subsequent papers have focussed on simpler Model 2 variants (Wang and Waibel, 1997), syntactic models (Wu, 1996), and cases without word-order variation (Tillmann *et al.*, 1997).

We now translate some novel sentences not seen in training.

```
ORIGINAL:
la empresa tiene enemigos fuertes en Europa .

AUTOMATIC TRANSLATIONS:
the company has strong enemies in Europe . 8e-08
the company has strong enemies in Europe . 2e-08
the company has strong enemies in Europe . 1e-08
the company has strong enemies in Europe . 8e-09
the in Europe company has strong enemies . 5e-09
the company has strong enemies in Europe . 5e-09
the company has strong enemies in Europe . 4e-09
the enemies in Europe company has strong . 4e-09
```

Notice that the k-best list includes repetitions. This is because we search for the best paths rather than the best sequences. Our smoothed language model

[2] **Implementation note.** We follow (Brown *et al.*, 1993) in Viterbi-style training, as the EM algorithm is too expensive even for this small corpus. We bootstrap initial parameters using Models 1 and 2. Our initial Model 3 training employs several simplifications: (1) we initialize fertilities for e_i based on the lengths of the sentence pairs in which it occurs, without consulting the t or a tables, (2) we do not use NULL, (3) we only use the immediate alignment neighborhood around the Model 2 Viterbi alignment (no pegging), (4) our parameters are stored in two-level hash tables in LISP.

On the other hand, our finite-state tools are reasonably complete. Our C++ implementation includes composition of weighted transducers and acceptors (Pereira and Riley, 1997) with full handling of ϵ-transitions. It includes forward-backward learning, which we use here to train and smooth our trigram language model. It also includes a k-best path extractor that follows the recently discovered $\mathrm{O}(m + n \log n + kn)$ algorithm of (Eppstein, 1994). This toolkit has been used for several MT-related tasks and is available for research purposes.

offers several paths that produce the same e, and we have already seen the multiplicity of paths relating a particular pair e and f. We typically ask for a large k and compress the result by summing identical sequence weights and resorting.

We can study the translation process more closely by inspecting the intermediate structures that get produced as our target sentence makes its way up the cascade. Our 8-word sentence first becomes a 256-state/1024-arc/40320-path permutation acceptor. Just before composition with the source language model, there are over 300 million English sentences (paths), stored in a 1537-state/3586-arc acceptor. The language model assigns low probability to most of these sentences, but does not actually filter any of them out.

Here are two more translations:

```
ORIGINAL:
sus grupos pequenos no venden medicinas .

AUTOMATIC TRANSLATIONS:
its small groups do not sell pharmaceuticals .
its small groups sell pharmaceuticals not .
its small groups sell not pharmaceuticals .
its small groups do not sell pharmaceuticals .
its small groups do not sell pharmaceuticals .
its not small groups sell pharmaceuticals .
its small groups sell pharmaceuticals do not .
its small groups sell pharmaceuticals not .
```

The composition automatically inserts the zero-fertility word *do* in every position of every permutation, and this proves useful to the source language model.

```
ORIGINAL:
los asociados estan enfadados con Carlos .

AUTOMATIC TRANSLATIONS:
the associates are angry with Carlos .
the associates are angry with Carlos .
the associates are with Carlos angry .
the Carlos associates are also angry .
Carlos the associates are also angry .
the Carlos Garcia associates are angry .
the Carlos has associates are angry .
the Carlos is associates are angry .
```

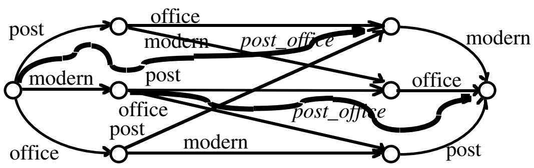
In this translation, the Spanish word *con* is unknown, never seen in training data. This would normally block the finite-state composition, but we have augmented our word-translation transducer to allow *any* English word to produce *con* (at low probability). The source language model uses context to select the translation *with*.

8 Better Finite-State Models

The previous section did not employ distortion probabilities. We can implement a $d(j|i)$ table by adding more transducers to the cascade. Source words are followed by source position indicators (like *spos-6*) that are carried along during the fertility and translation stages. Later, they are probabilistically converted to target position indicators (like *tpos-5*). Permutation acceptors must include target positions after every word, to match the output of the $d(j|i)$ transducer.

We can also address some problems with Model 3. For example, a word's different fertilities are not tied to its different translations. When used as a noun, *slap* has a fertility of only one; and it never translates to *daba*, only to *botefada*. While this information is not available from the probability tables, it is available from the bank of most-probable alignments. We could revise our fertility transducer to replace *slap* with either *slap-1* or *slap-3* (instead of triplicating the word). Then our word translation device would convert *slap-3* into a sequence of three target words, and *slap-1* would simply go to *botefada*.

Model 3 is also loose about phrasal translation. If *una*, *daba*, and *botefada* are spread widely within a sentence, *slap* still becomes a candidate source word. It may be clear from inspecting the alignments, however that these three words are always produced adjacent to one another. In that case, our word translation device could produce a token like *daba_una_bofetada*. This would incidentally account for order within a phrase—Model 3 does not distinguish between orderings like *modern post office* and *post modern office*. We must add information to the permutation acceptor, e.g., by adding arcs for known phrases:



Tokens like *post_office* and *daba_una_bofetada* would then incur single rather than multiple distortion costs. This last consideration motivates Models 4 and 5 of (Brown *et al.*, 1993).

9 Discussion

We have presented a reconstruction of statistical machine translation in terms of cascaded finite-state automata, and we have shown how these automata can be used in translating sentences and calculating conditional probabilities. We use general composition and decoding algorithms that are also useful in a wide range of other applications.

There are several interesting questions. First, will the finite-state composition scale up to large vocabularies and sentence lengths? (Brown *et al.*, 1993) employ an undocumented suboptimal search and report search errors, so scale is clearly a problem. On the other hand, finite-state cascades have been shown to greatly reduce memory requirements for speech recognition, and there now appear to be pruning and lazy composition methods that find approximate best paths in the face of overwhelming possibilities. It is interesting to consider the development of general methods versus MT-specific solutions. Of course, if MT-specific knowledge could reduce the size of the permutation acceptors, that would help greatly.

Another question is: can finite-state methods be used in model training as well as decoding? Forward-backward training is problematic for a couple of reasons. One is the existence of many local minima. A second is the fact that training consists of input sequences paired with output permutation acceptors (not sequences).

However, it may be practical to use finite-state cascades to compute $P(f|e)$ during training iterations. Further, it may be possible to use k-best lists to locate a subset of alignments for which to collect parameter counts at each iteration. This could call into question the idea of summing over alignments—why not sum over paths?

Of course, alignments are useful for other things (such as sense-tagging), but when transducer composition employs a judicious naming scheme for result-states, alignments are recoverable from paths.

Finally, it is interesting to consider the probabilistic integration of speech and character-recognition—or of any other process that presents ambiguity, such as part-of-speech tagging, sense tagging, morphological analysis, accent restoration, and spelling correction.

References

H. Alshawi, A. Buchsbaum, and F. Xia. 1997. A comparison of head transducers and transfer for a limited domain translation applications. In *Proc. ACL*.

P. F. Brown, S. A. Della-Pietra, V. J. Della-Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2).

D. Eppstein. 1994. Finding the k shortest paths. In *Proc. 35th FOCS*.

F. Jelinek and R. L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Pattern Recognition in Practice*. North-Holland, Amsterdam.

K. Knight. 1997. Automating knowledge acquisition for machine translation. *AI Magazine*, 18(4).

F. Pereira and M. Riley. 1997. Speech recognition by composition of weighted finite automata. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*. MIT Press.

E. Roche and Y. Schabes, editors. 1997. *Finite-State Language Processing*. MIT Press.

K.-Y. Su, J.-S. Chang, and Y.-L. U. Hsu. 1995. A corpus-based two-way design for parameterized MT systems: Rationale, architecture and training issues. In *Proc. TMI*.

C. Tillmann, S. Vogel, H. Ney, and A. Zubiaga. 1997. A DP-based search using monotone alignments in statistical translation. In *Proc. ACL*.

Y. Wang and A. Waibel. 1997. Decoding algorithm in statistical machine translation. In *Proc. ACL*.

D. Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. ACL*.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3).