A Syntax-based Statistical Translation Model

Kenji Yamada and **Kevin Knight**

Information Sciences Institute

University of Southern California

4676 Admiralty Way, Suite 1001

Marina del Rey, CA 90292

{kyamada,knight}@isi.edu

Abstract

We present a syntax-based statistical translation model. Our model transforms a source-language parse tree into a target-language string by applying stochastic operations at each node. These operations capture linguistic differences such as word order and case marking. Model parameters are estimated in polynomial time using an EM algorithm. The model produces word alignments that are better than those produced by IBM Model 5.

1 Introduction

A statistical translation model (TM) is a mathematical model in which the process of human-language translation is statistically modeled. Model parameters are automatically estimated using a corpus of translation pairs. TMs have been used for statistical machine translation (Berger et al., 1996), word alignment of a translation corpus (Melamed, 2000), multilingual document retrieval (Franz et al., 1999), automatic dictionary construction (Resnik and Melamed, 1997), and data preparation for word sense disambiguation programs (Brown et al., 1991). Developing a better TM is a fundamental issue for those applications.

Researchers at IBM first described such a statistical TM in (Brown et al., 1988). Their models are based on a string-to-string noisy channel model. The channel converts a sequence of words in one language (such as English) into another (such as French). The channel operations are movements, duplications, and translations, applied to each word independently. The movement

is conditioned only on word classes and positions in the string, and the duplication and translation are conditioned only on the word identity. Mathematical details are fully described in (Brown et al., 1993).

One criticism of the IBM-style TM is that it does not model structural or syntactic aspects of the language. The TM was only demonstrated for a structurally similar language pair (English and French). It has been suspected that a language pair with very different word order such as English and Japanese would not be modeled well by these TMs.

To incorporate structural aspects of the language, our channel model accepts a parse tree as an input, i.e., the input sentence is preprocessed by a syntactic parser. The channel performs operations on each node of the parse tree. The operations are *reordering* child nodes, *inserting* extra words at each node, and *translating* leaf words. Figure 1 shows the overview of the operations of our model. Note that the output of our model is a string, not a parse tree. Therefore, parsing is only needed on the channel input side.

The reorder operation is intended to model translation between languages with different word orders, such as SVO-languages (English or Chinese) and SOV-languages (Japanese or Turkish). The word-insertion operation is intended to capture linguistic differences in specifying syntactic cases. E.g., English and French use structural position to specify case, while Japanese and Korean use case-marker particles.

Wang (1998) enhanced the IBM models by introducing phrases, and Och *et al.* (1999) used templates to capture phrasal sequences in a sentence. Both also tried to incorporate structural aspects of the language, however, neither handles
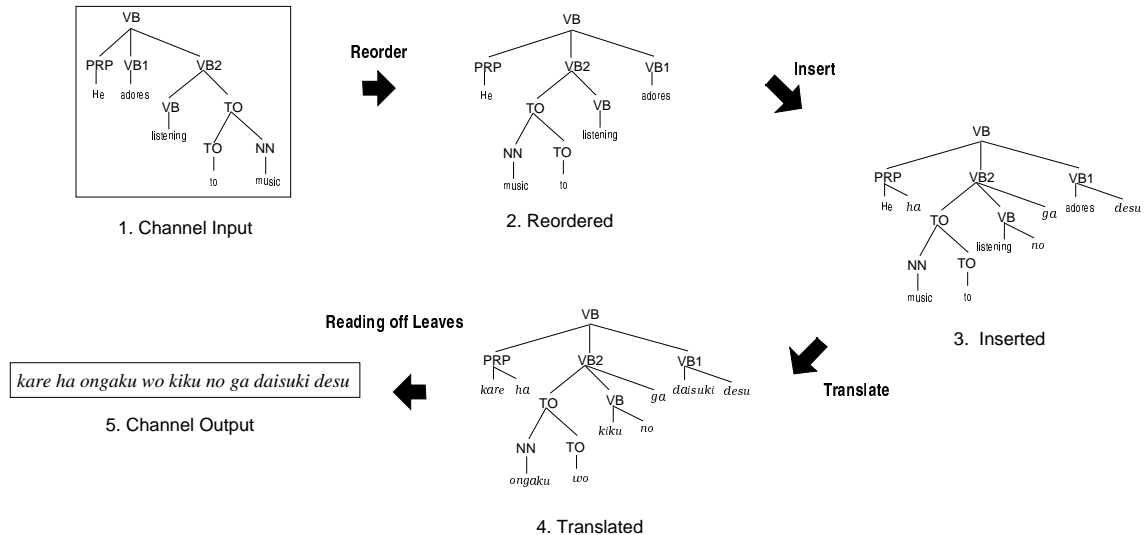
Figure 1: Channel Operations: Reorder, Insert, and Translate

nested structures.

Wu (1997) and Alshawi *et al.* (2000) showed statistical models based on syntactic structure. The way we handle syntactic parse trees is inspired by their work, although their approach is not to model the translation process, but to formalize a model that generates two languages at the same time. Our channel operations are also similar to the mechanism in Twisted Pair Grammar (Jones and Havrilla, 1998) used in their knowledge-based system.

Following (Brown et al., 1993) and the other literature in TM, this paper only focuses the details of TM. Applications of our TM, such as machine translation or dictionary construction, will be described in a separate paper. Section 2 describes our model in detail. Section 3 shows experimental results. We conclude with Section 4, followed by an Appendix describing the training algorithm in more detail.

2 The Model

2.1 An Example

We first introduce our translation model with an example. Section 2.2 will describe the model more formally. We assume that an English parse tree is fed into a noisy channel and that it is translated to a Japanese sentence.[1]

---

[1] The parse tree is flattened to work well with the model. See Section 3.1 for details.

Figure 1 shows how the channel works. First, child nodes on each internal node are stochastically *reordered*. A node with N children has $N!$ possible reorderings. The probability of taking a specific reordering is given by the model's *r-table*. Sample model parameters are shown in Table 1. We assume that only the sequence of child node labels influences the reordering. In Figure 1, the top VB node has a child sequence PRP-VB1-VB2. The probability of reordering it into PRP-VB2-VB1 is 0.723 (the second row in the r-table in Table 1). We also reorder VB-TO into TO-VB, and TO-NN into NN-TO, so therefore the probability of the second tree in Figure 1 is $0.723 \cdot 0.749 \cdot 0.893 = 0.484$.

Next, an extra word is stochastically *inserted* at each node. A word can be inserted either to the left of the node, to the right of the node, or nowhere. Brown *et al.* (1993) assumes that there is an invisible NULL word in the input sentence and it generates output words that are distributed into random positions. Here, we instead decide the position on the basis of the nodes of the input parse tree. The insertion probability is determined by the *n-table*. For simplicity, we split the n-table into two: a table for insert positions and a table for words to be inserted (Table 1). The node's label and its parent's label are used to index the table for insert positions. For example, the PRP node in Figure 1 has parent VB, thus

original order	reordering	P(reorder)
PRP VB1 VB2	PRP VB1 VB2	0.074
	PRP VB2 VB1	0.723
	VB1 PRP VB2	0.061
	VB1 VB2 PRP	0.037
	VB2 PRP VB1	0.083
	VB2 VB1 PRP	0.021
VB TO	VB TO	0.251
	TO VB	0.749
TO NN	TO NN	0.107
	NN TO	0.893
⋮	⋮	⋮

r-table

parent	TOP	VB	VB	VB	TO	TO	...
node	VB	VB	PRP	TO	TO	NN	...
P(None)	0.735	0.687	0.344	0.709	0.900	0.800	...
P(Left)	0.004	0.061	0.004	0.030	0.003	0.096	...
P(Right)	0.260	0.252	0.652	0.261	0.007	0.104	...

n–table

w	P(ins-w)
ha	0.219
ta	0.131
wo	0.099
no	0.094
ni	0.080
te	0.078
ga	0.062
desu	0.0007
⋮	

E	adores		he		i		listening		music		to		...
J	daisuki	1.000	kare	0.952	NULL	0.471	kiku	0.333	ongaku	0.900	ni	0.216	...
			NULL	0.016	watasi	0.111	kii	0.333	naru	0.100	NULL	0.204	
			nani	0.005	kare	0.055	mi	0.333			to	0.133	
			da	0.003	shi	0.021					no	0.046	
			shi	0.003	nani	0.020					wo	0.038	
			⋮		⋮		⋮		⋮		⋮		

t–table

Table 1: Model Parameter Tables

⟨parent=VB, node=PRP⟩ is the conditioning index. Using this label pair captures, for example, the regularity of inserting case-marker particles. When we decide which word to insert, no conditioning variable is used. That is, a function word like *ga* is just as likely to be inserted in one place as any other. In Figure 1, we inserted four words (*ha, no, ga* and *desu*) to create the third tree. The top VB node, two TO nodes, and the NN node inserted nothing. Therefore, the probability of obtaining the third tree given the second tree is $(0.652 \cdot 0.219) \cdot (0.252 \cdot 0.094) \cdot (0.252 \cdot 0.062) \cdot (0.252 \cdot 0.0007) \cdot 0.735 \cdot 0.709 \cdot 0.900 \cdot 0.800 = 3.498e\text{-}9$.

Finally, we apply the *translate* operation to each leaf. We assume that this operation is dependent only on the word itself and that no context is consulted.[2] The model's *t-table* specifies the probability for all cases. Suppose we obtained the translations shown in the fourth tree of Figure 1. The probability of the translate operation here is $0.952 \cdot 0.900 \cdot 0.038 \cdot 0.333 \cdot 1.000 = 0.0108$.

The total probability of the *reorder*, *insert* and *translate* operations in this example is $0.484 \cdot 3.498e\text{-}9 \cdot 0.0108 = 1.828e\text{-}11$. Note that there

are many other combinations of such operations that yield the same Japanese sentence. Therefore, the probability of the Japanese sentence given the English parse tree is the sum of all these probabilities.

We actually obtained the probability tables (Table 1) from a corpus of about two thousand pairs of English parse trees and Japanese sentences, completely automatically. Section 2.3 and Appendix 4 describe the training algorithm.

2.2 Formal Description

This section formally describes our translation model. To make this paper comparable to (Brown et al., 1993), we use English-French notation in this section. We assume that an English parse tree $\mathcal{E}$ is transformed into a French sentence **f**. Let the English parse tree $\mathcal{E}$ consist of nodes $\varepsilon_1, \varepsilon_2 \ldots, \varepsilon_n$, and let the output French sentence consist of French words $f_1, f_2, \ldots, f_m$.

Three random variables, **N**, **R**, and **T** are channel operations applied to each node. *Insertion* **N** is an operation that inserts a French word just before or after the node. The insertion can be *none*, *left*, or *right*. Also it decides what French word to insert. *Reorder* **R** is an operation that changes the order of the children of the node. If a node has three children, e.g., there are $3! = 6$ ways

---

[2] When a TM is used in machine translation, the TM's role is to provide a list of possible translations, and a language model addresses the context. See (Berger et al., 1996).

to reorder them. This operation applies only to non-terminal nodes in the tree. *Translation* **T** is an operation that translates a terminal English leaf word into a French word. This operation applies only to terminal nodes. Note that an English word can be translated into a French NULL word.

The notation $\theta = \langle \nu, \rho, \tau \rangle$ stands for a set of values of $\langle \mathbf{N}, \mathbf{R}, \mathbf{T} \rangle$. $\theta_i = \langle \nu_i, \rho_i, \tau_i \rangle$ is a set of values of random variables associated with ε_i. And $\boldsymbol{\theta} = \theta_1, \theta_2, \ldots, \theta_n$ is the set of all random variables associated with a parse tree $\mathcal{E} = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$.

The probability of getting a French sentence $\mathbf{f}$ given an English parse tree $\mathcal{E}$ is

$$\mathrm{P}(\mathbf{f}|\mathcal{E}) = \sum_{\boldsymbol{\theta}:\mathrm{Str}(\boldsymbol{\theta}(\mathcal{E}))=\mathbf{f}} \mathrm{P}(\boldsymbol{\theta}|\mathcal{E})$$

where $\mathrm{Str}(\boldsymbol{\theta}(\mathcal{E}))$ is the sequence of leaf words of a tree transformed by $\boldsymbol{\theta}$ from $\mathcal{E}$.

The probability of having a particular set of values of random variables in a parse tree is

$$
\begin{aligned}
\mathrm{P}(\boldsymbol{\theta}|\mathcal{E}) &= \mathrm{P}(\theta_1, \theta_2, \ldots, \theta_n | \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n) \\
&= \prod_{i=1}^{n} \mathrm{P}(\theta_i | \theta_1, \theta_2, \ldots, \theta_{i-1}, \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n).
\end{aligned}
$$

This is an exact equation. Then, we assume that a transform operation is independent from other transform operations, and the random variables of each node are determined only by the node itself. So, we obtain

$$
\begin{aligned}
\mathrm{P}(\boldsymbol{\theta}|\mathcal{E}) &= \mathrm{P}(\theta_1, \theta_2, \ldots, \theta_n | \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n) \\
&= \prod_{i=1}^{n} \mathrm{P}(\theta_i | \varepsilon_i).
\end{aligned}
$$

The random variables $\theta_i = \langle \nu_i, \rho_i, \tau_i \rangle$ are assumed to be independent of each other. We also assume that they are dependent on particular features of the node ε_i. Then,

$$
\begin{aligned}
\mathrm{P}(\theta_i|\varepsilon_i) &= \mathrm{P}(\nu_i, \rho_i, \tau_i | \varepsilon_i) \\
&= \mathrm{P}(\nu_i|\varepsilon_i)\,\mathrm{P}(\rho_i|\varepsilon_i)\,\mathrm{P}(\tau_i|\varepsilon_i) \\
&= \mathrm{P}(\nu_i|\mathcal{N}(\varepsilon_i))\,\mathrm{P}(\rho_i|\mathcal{R}(\varepsilon_i))\,\mathrm{P}(\tau_i|\mathcal{T}(\varepsilon_i)) \\
&= n(\nu_i|\mathcal{N}(\varepsilon_i))\,r(\rho_i|\mathcal{R}(\varepsilon_i))\,t(\tau_i|\mathcal{T}(\varepsilon_i))
\end{aligned}
$$

where $\mathcal{N}$, $\mathcal{R}$, and $\mathcal{T}$ are the relevant features to **N**, **R**, and **T**, respectively. For example, we saw that the parent node label and the node label were used for $\mathcal{N}$, and the syntactic category sequence

of children was used for $\mathcal{R}$. The last line in the above formula introduces a change in notation, meaning that those probabilities are the model parameters $n(\nu|\mathcal{N})$, $r(\rho|\mathcal{R})$, and $t(\tau|\mathcal{T})$, where $\mathcal{N}$, $\mathcal{R}$, and $\mathcal{T}$ are the possible values for $\mathcal{N}$, $\mathcal{R}$, and $\mathcal{T}$, respectively.

In summary, the probability of getting a French sentence $\mathbf{f}$ given an English parse tree $\mathcal{E}$ is

$$\mathrm{P}(\mathbf{f}|\mathcal{E}) = \sum_{\boldsymbol{\theta}:\mathrm{Str}(\boldsymbol{\theta}(\mathcal{E}))=\mathbf{f}} \mathrm{P}(\boldsymbol{\theta}|\mathcal{E})$$

$$= \sum_{\boldsymbol{\theta}:\mathrm{Str}(\boldsymbol{\theta}(\mathcal{E}))=\mathbf{f}} \prod_{i=1}^{n} n(\nu_i|\mathcal{N}(\varepsilon_i))\,r(\rho_i|\mathcal{R}(\varepsilon_i))\,t(\tau_i|\mathcal{T}(\varepsilon_i))$$

where $\mathcal{E} = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ and $\boldsymbol{\theta} = \theta_1, \theta_2, \ldots, \theta_n = \langle \nu_1, \rho_1, \tau_1 \rangle, \langle \nu_2, \rho_2, \tau_2 \rangle, \ldots, \langle \nu_n, \rho_n, \tau_n \rangle$.

The model parameters $n(\nu|\mathcal{N})$, $r(\rho|\mathcal{R})$, and $t(\tau|\mathcal{T})$, that is, the probabilities $\mathrm{P}(\nu|\mathcal{N})$, $\mathrm{P}(\rho|\mathcal{R})$ and $\mathrm{P}(\tau|\mathcal{T})$, decide the behavior of the translation model, and these are the probabilities we want to estimate from a training corpus.

2.3 Automatic Parameter Estimation

To estimate the model parameters, we use the EM algorithm (Dempster et al., 1977). The algorithm iteratively updates the model parameters to maximize the likelihood of the training corpus. First, the model parameters are initialized. We used a uniform distribution, but it can be a distribution taken from other models. For each iteration, the number of events are counted and weighted by the probabilities of the events. The probabilities of events are calculated from the current model parameters. The model parameters are re-estimated based on the counts, and used for the next iteration. In our case, an event is a pair of a value of a random variable (such as ν, ρ, or τ) and a feature value (such as $\mathcal{N}$, $\mathcal{R}$, or $\mathcal{T}$). A separate counter is used for each event. Therefore, we need the same number of counters, $c(\nu, \mathcal{N})$, $c(\rho, \mathcal{R})$, and $c(\tau, \mathcal{T})$, as the number of entries in the probability tables, $n(\nu|\mathcal{N})$, $r(\rho|\mathcal{R})$, and $t(\tau|\mathcal{T})$.

The training procedure is the following:

1. Initialize all probability tables: $n(\nu|\mathcal{N})$, $r(\rho|\mathcal{R})$, and $t(\tau|\mathcal{T})$.

2. Reset all counters: $c(\nu, \mathcal{N})$, $c(\rho, \mathcal{R})$, and $c(\tau, \mathcal{T})$.

3. For each pair $\langle \mathcal{E}, \mathbf{f} \rangle$ in the training corpus,

   For all $\boldsymbol{\theta}$, such that $\mathbf{f} = \mathrm{Str}(\boldsymbol{\theta}(\mathcal{E}))$,
   - Let cnt $= \mathrm{P}(\boldsymbol{\theta}|\mathcal{E}) / \sum_{\boldsymbol{\theta}:\mathrm{Str}(\boldsymbol{\theta}(\mathcal{E}))=\mathbf{f}} \mathrm{P}(\boldsymbol{\theta}|\mathcal{E})$

- For $i = 1 \ldots n$,
  $c(\nu_i, \mathcal{N}(\varepsilon_i))$ += cnt
  $c(\rho_i, \mathcal{R}(\varepsilon_i))$ += cnt
  $c(\tau_i, \mathcal{T}(\varepsilon_i))$ += cnt

4. For each $\langle \nu, \mathcal{N} \rangle$, $\langle \rho, \mathcal{R} \rangle$, and $\langle \tau, \tau \rangle$,

$$n(\nu|\mathcal{N}) = c(\nu, \mathcal{N}) / \sum_\nu c(\nu, \mathcal{N})$$
$$r(\rho|\mathcal{R}) = c(\rho, \mathcal{R}) / \sum_\rho c(\rho, \mathcal{R})$$
$$t(\tau|\tau) = c(\tau, \tau) / \sum_\tau c(\tau, \tau)$$

5. Repeat steps 2-4 for several iterations.

A straightforward implementation that tries all possible combinations of parameters $\langle \nu, \rho, \tau \rangle$, is very expensive, since there are $O(|\nu|^n |\rho|^n)$ possible combinations, where $|\nu|$ and $|\rho|$ are the number of possible values for ν and ρ, respectively (τ is uniquely decided when ν and ρ are given for a particular $\langle \mathcal{E}, \mathbf{f} \rangle$). Appendix describes an efficient implementation that estimates the probability in polynomial time.[3] With this efficient implementation, it took about 50 minutes per iteration on our corpus (about two thousand pairs of English parse trees and Japanese sentences. See the next section).

3 Experiment

To experiment, we trained our model on a small English-Japanese corpus. To evaluate performance, we examined alignments produced by the learned model. For comparison, we also trained IBM Model 5 on the same corpus.

3.1 Training

We extracted 2121 translation sentence pairs from a Japanese-English dictionary. These sentences were mostly short ones. The average sentence length was 6.9 for English and 9.7 for Japanese. However, many rare words were used, which made the task difficult. The vocabulary size was 3463 tokens for English, and 3983 tokens for Japanese, with 2029 tokens for English and 2507 tokens for Japanese occurring only once in the corpus.

Brill's part-of-speech (POS) tagger (Brill, 1995) and Collins' parser (Collins, 1999) were used to obtain parse trees for the English side of the corpus. The output of Collins' parser was

---

[3]Note that the algorithm performs full EM counting, whereas the IBM models only permit counting over a subset of possible alignments.

modified in the following way. First, to reduce the number of parameters in the model, each node was re-labelled with the POS of the node's head word, and some POS labels were collapsed. For example, labels for different verb endings (such as VBD for -ed and VBG for -ing) were changed to the same label VB. There were then 30 different node labels, and 474 unique child label sequences.

Second, a subtree was flattened if the node's head-word was the same as the parent's headword. For example, (NN1 (VB NN2)) was flattened to (NN1 VB NN2) if the VB was a head word for both NN1 and NN2. This flattening was motivated by various word orders in different languages. An English SVO structure is translated into SOV in Japanese, or into VSO in Arabic. These differences are easily modeled by the flattened subtree (NN1 VB NN2), rather than (NN1 (VB NN2)).

We ran 20 iterations of the EM algorithm as described in Section 2.2. IBM Model 5 was sequentially bootstrapped with Model 1, an HMM Model, and Model 3 (Och and Ney, 2000). Each preceding model and the final Model 5 were trained with five iterations (total 20 iterations).

3.2 Evaluation

The training procedure resulted in the tables of estimated model parameters. Table 1 in Section 2.1 shows part of those parameters obtained by the training above.

To evaluate performance, we let the models generate the most probable alignment of the training corpus (called the Viterbi alignment). The alignment shows how the learned model induces the internal structure of the training data.

Figure 2 shows alignments produced by our model and IBM Model 5. Darker lines indicates that the particular alignment link was judged correct by humans. Three humans were asked to rate each alignment as *okay* (1.0 point), *not sure* (0.5 point), or *wrong* (0 point). The darkness of the lines in the figure reflects the human score. We obtained the average score of the first 50 sentence pairs in the corpus. We also counted the number of perfectly aligned sentence pairs in the 50 pairs. Perfect means that all alignments in a sentence pair were judged *okay* by all the human judges.
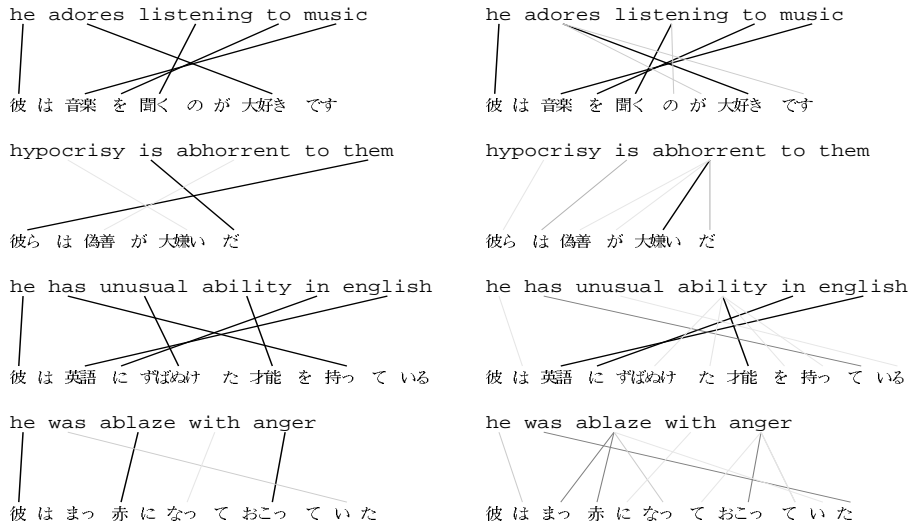
Figure 2: Viterbi Alignments: our model (left) and IBM Model 5 (right). Darker lines are judged more correct by humans.

The result was the following;

	Alignment ave. score	Perfect sents
Our Model	0.582	10
IBM Model 5	0.431	0

Our model got a better result compared to IBM Model 5. Note that there were no perfect alignments from the IBM Model. Errors by the IBM Model were spread out over the whole set, while our errors were localized to some sentences. We expect that our model will therefore be easier to improve. Also, localized errors are good if the TM is used for corpus preparation or filtering.

We also measured training perplexity of the models. The perplexity of our model was 15.79, and that of IBM Model 5 was 9.84. For reference, the perplexity after 5 iterations of Model 1 was 24.01. Perplexity values roughly indicate the predictive power of the model. Generally, lower perplexity means a better model, but it might cause over-fitting to a training data. Since the IBM Model usually requires millions of training sentences, the lower perplexity value for the IBM Model is likely due to over-fitting.

4 Conclusion

We have presented a syntax-based translation model that statistically models the translation process from an English parse tree into a foreign-language sentence. The model can make use of syntactic information and performs better for language pairs with different word orders and case marking schema. We conducted a small-scale experiment to compare the performance with IBM Model 5, and got better alignment results.

Appendix: An Efficient EM algorithm

This appendix describes an efficient implementation of the EM algorithm for our translation model. This implementation uses a graph structure for a pair $\langle \mathcal{E}, \mathbf{f} \rangle$. A graph node is either a *major-node* or a *subnode*. A major-node shows a pairing of a subtree of $\mathcal{E}$ and a substring of $\mathbf{f}$. A subnode shows a selection of a value $\langle \nu, \rho, \tau \rangle$ for the subtree-substring pair (Figure 3).

Let $\mathbf{f}_k^l = f_k \ldots f_{k+(l-1)}$ be a substring of $\mathbf{f}$ from the word f_k with length l. Note this notation is different from (Brown et al., 1993). A subtree ε_i is a subtree of $\mathcal{E}$ below the node ε_i. We assume that a subtree ε_1 is $\mathcal{E}$.

A *major-node* $v(\varepsilon_i, \mathbf{f}_k^l)$ is a pair of a subtree ε_i and a substring $\mathbf{f}_k^l$. The root of the graph is $v(\varepsilon_1, \mathbf{f}_1^L)$, where L is the length of $\mathbf{f}$. Each major-node connects to several ν-*subnodes* $v(\nu; \varepsilon_i, \mathbf{f}_k^l)$, showing which value of ν is selected. The arc between $v(\varepsilon_i, \mathbf{f}_k^l)$ and $v(\nu; \varepsilon_i, \mathbf{f}_k^l)$ has weight $\mathrm{P}(\nu|\varepsilon_i)$.

A ν-subnode $v(\nu; \varepsilon_i, \mathbf{f}_k^l)$ connects to a *final-node* with weight $\mathrm{P}(\tau|\varepsilon_i)$ if ε_i is a terminal node

in $\mathcal{E}$. If ε_i is a non-terminal node, a ν-subnode connects to several ρ-*subnodes* $v(\rho; \nu, \varepsilon_i, \mathbf{f}_k^l)$, showing a selection of a value ρ. The weight of the arc is $P(\rho|\varepsilon_i)$.

A ρ-subnode is then connected to π-subnodes $v(\pi; \rho, \nu, \varepsilon_i, \mathbf{f}_k^l)$. The *partition* variable, π, shows a particular way of partitioning $\mathbf{f}_k^l$.

A π-subnode $v(\pi; \rho, \nu, \varepsilon_i, \mathbf{f}_k^l)$ is then connected to major-nodes which correspond to the children of ε_i and the substring of $\mathbf{f}_k^l$, decided by $\langle \nu, \rho, \pi \rangle$. A major-node can be connected from different π-subnodes. The arc weights between ρ-subnodes and major-nodes are always 1.0.
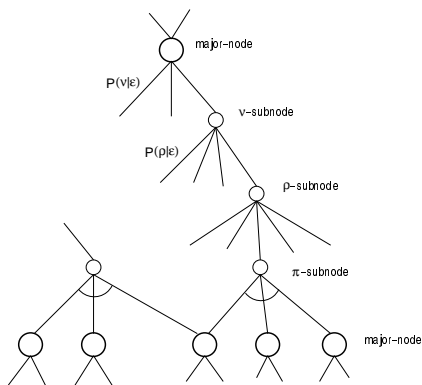

Figure 3: Graph structure for efficient EM training.

This graph structure makes it easy to obtain $P(\boldsymbol{\theta}|\mathcal{E})$ for a particular $\boldsymbol{\theta}$ and $\sum_{\boldsymbol{\theta}:\mathrm{Str}(\boldsymbol{\theta}(\mathcal{E}))=\mathbf{f}} P(\boldsymbol{\theta}|\mathcal{E})$. A trace starting from the graph root, selecting one of the arcs from major-nodes, ν-subnodes, and ρ-subnodes, and *all* the arcs from π-subnodes, corresponds to a particular $\boldsymbol{\theta}$, and the product of the weight on the trace corresponds to $P(\boldsymbol{\theta}|\mathcal{E})$. Note that a trace forms a tree, making branches at the π-subnodes.

We define an alpha probability and a beta probability for each major-node, in analogy with the measures used in the inside-outside algorithm for probabilistic context free grammars (Baker, 1979).

The alpha probability (outside probability) is a path probability from the graph root to the node and the side branches of the node. The beta probability (inside probability) is a path probability below the node.

Figure 4 shows formulae for alpha-beta probabilities. From these definitions,

$$\sum_{\boldsymbol{\theta}:\mathrm{Str}(\boldsymbol{\theta}(\mathcal{E}))=\mathbf{f}} P(\boldsymbol{\theta}|\mathcal{E}) = \beta(\varepsilon_1, \mathbf{f}_1^L) \;.$$

The counts $c(\nu, \varkappa)$, $c(\rho, \varkappa)$, and $c(\tau, \tau)$ for each pair $\langle \mathcal{E}, \mathbf{f} \rangle$ are also in the figure. Those formulae replace the step 3 (in Section 2.3) for each training pair, and these counts are used in the step 4.

The graph structure is generated by expanding the root node $v(\varepsilon_1, \mathbf{f}_1^L)$. The beta probability for each node is first calculated bottom-up, then the alpha probability for each node is calculated top-down. Once the alpha and beta probabilities for each node are obtained, the counts are calculated as above and used for updating the parameters.

The complexity of this training algorithm is $O(n^3|\nu||\rho||\pi|)$. The cube comes from the number of parse tree nodes (n) and the number of possible French substrings (n^2).

Acknowledgments

References

H. Alshawi, S. Bangalore, and S. Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26(1).

J. Baker. 1979. Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Sciety of America*.

A. Berger, P. Brown, S. Della Pietra, V. Della Pietra, J. Gillett, J. Lafferty, R. Mercer, H. Printz, and L. Ures. 1996. *Language Translation Apparatus and Method Using Context-Based Translation Models*. U.S. Patent 5,510,981.

E. Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of specch tagging. *Computational Linguistics*, 21(4).

P. Brown, J. Cocke, S. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. 1988. A statistical approach to language translation. In *COLING-88*.

P. Brown, J. Cocke, S. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. 1991. Word-sense disambiguation using statistical methods. In *ACL-91*.

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

The alpha probability for the graph root, $\alpha(\varepsilon_1, \mathbf{f}_1^L)$, is 1.0. For other major-nodes,

$$\alpha(v) = \sum_{s \,\in\, \text{Parent}_\pi(v)} \alpha(\text{Parent}_M(s)) \cdot \Big\{ \text{P}(\nu|\varepsilon)\text{P}(\rho|\varepsilon) \prod_{\substack{v' \,\in\, \text{Child}_\pi(s) \\ v' \neq v}} \beta(v') \Big\}$$

where $\text{Parent}_\pi(v)$ is a set of π-subnodes which are immediate parents of v, $\text{Child}_\pi(s)$ is a set of major-nodes which are children of π-subnodes s, and $\text{Parent}_M(s)$ is a parent major-node of π-subnodes s (skipping ρ-subnodes, and ν-subnodes). $\text{P}(\nu|\varepsilon)$ and $\text{P}(\rho|\varepsilon)$ are the arc weights from $\text{Parent}_M(s)$ to s.

The beta probability is defined as

$$\begin{aligned}
\beta(v) &= \beta(\varepsilon_i, \mathbf{f}_k^l) \\
&= \begin{cases} \text{P}(\tau|\varepsilon_i) & \text{if } \varepsilon_i \text{ is a terminal} \\ \sum_\nu \text{P}(\nu|\varepsilon_i) \sum_\rho \text{P}(\rho|\varepsilon_i) \sum_\pi \prod_j \beta(\varepsilon_j, \mathbf{f}_{k'}^{l'}) & \text{if } \varepsilon_i \text{ is a non-terminal} \end{cases}
\end{aligned}$$

where ε_j is a child of ε_i, and $\mathbf{f}_{k'}^{l'}$ is a proper partition of $\mathbf{f}_k^l$.

The counts $c(\nu, \mathcal{N})$, $c(\rho, \mathcal{R})$, and $c(\tau, \tau)$ for each pair $\langle \mathcal{E}, \mathbf{f} \rangle$ are,

$$c(\nu, \mathcal{N}) = \sum_{\substack{k,l \\ \varepsilon_i \,:\, \mathcal{N}(\varepsilon_i) = \mathcal{N}}} \Big\{ \alpha(\varepsilon_i, \mathbf{f}_k^l)\text{P}(\nu|\varepsilon_i) \sum_\rho \text{P}(\rho|\varepsilon_i) \sum_\pi \prod_j \beta(\varepsilon_j, \mathbf{f}_{k'}^{l'}) \Big\} \Big/ \beta(\varepsilon_1, \mathbf{f}_1^L)$$

$$c(\rho, \mathcal{R}) = \sum_{\substack{k,l \\ \varepsilon_i \,:\, \mathcal{R}(\varepsilon_i) = \mathcal{R}}} \Big\{ \alpha(\varepsilon_i, \mathbf{f}_k^l)\text{P}(\rho|\varepsilon_i) \sum_\nu \text{P}(\nu|\varepsilon_i) \sum_\pi \prod_j \beta(\varepsilon_j, \mathbf{f}_{k'}^{l'}) \Big\} \Big/ \beta(\varepsilon_1, \mathbf{f}_1^L)$$

$$c(\tau, \tau) = \sum_{\substack{k,l \\ \varepsilon_i \,:\, \mathcal{T}(\varepsilon_i) = \tau}} \Big\{ \alpha(\varepsilon_i, \mathbf{f}_k^l)\text{P}(\tau|\varepsilon_i) \Big\} \Big/ \beta(\varepsilon_1, \mathbf{f}_1^L)$$

where $1 \leq k \leq (L+1)$, $0 \leq l \leq L$, and L is the length of $\mathbf{f}$, since an English word can match a NULL French word.

Figure 4: Formulae for alpha-beta probabilities, and the count derivation

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Royal Statistical Society Series B*, 39.

M. Franz, J. McCarley, and R. Ward. 1999. Ad hoc, cross-language and spoken document information retrieval at IBM. In *TREC-8*.

D. Jones and R. Havrilla. 1998. Twisted pair grammar: Support for rapid development of machine translation for low density languages. In *AMTA98*.

I. Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2).

F. Och and H. Ney. 2000. Improved statistical alignment models. In *ACL-2000*.

F. Och, C. Tillmann, and H. Ney. 1999. Improved alignment models for statistical machine translation. In *EMNLP-99*.

P. Resnik and I. Melamed. 1997. Semi-automatic acquisition of domain-specific translation lexicons. In *ANLP-97*.

Y. Wang. 1998. *Grammar Inference and Statistical Machine Translation*. Ph.D. thesis, Carnegie Mellon University.

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