CMPT 825: Natural Language Processing **Spring 2006**

Lecture 8 — Mar 12, 2008

Lecturer: Anoop Sarkar *Scribe: Mohammad Norouzi*

8.1 Statistical Machine Translation

In this lecture we continue the study of machine translation in more details. In the previous lectrue, the translation model 3 was introduced. We will explain four other translation models and compare them with model 3. We also describe a computationally cheap learning algorithm for these models, given a set of bilingual texts. at the end, a HMM-based alignment model will be discussed briefly.

8.2 Quick Review

As usual, we consider translation from source language F to the target language E, let's say French to English. Using base rule, we can rewrite $p(e|f)$ as $p(e)p(f|e)$. Brown et al. [1] introduced word-by-word alignment between pairs of sentences f and e. One can think of alignment in different ways. We focus on many-to-one alignment from f to e and vector a is used to represent this alignment. We assume e has length l and f has length m, so the size of a is also m. $a_j = i$ means that f_j; the jth word of f is associated with e_i; the ith word of e.

To learn from pairs of translated sentences we should have some idea about the alignment of French and English words. We can consider all possible alignments and assign appropriate probabilities to them to accordingly compute other parameters of the model. Given sentences f and e, we should compute the probability of a specific alignment a.

$$p(a|f,e) = \frac{p(a,f|e)}{p(f|e)} = \frac{p(a,f|e)}{\sum_{a'} p(a',f|e)} \tag{8.1}$$

Therefore, all of our effort in translation models would be to compute the probability $p(a,f|e)$ that leads to the computation of $p(f|e)$ and $p(a|f,e)$.

If we allow `null` word on the English side, there are $(l+1)^m$ acceptable alignments. Obviously, summing over all alignments is a computationally expensive task. Thus, we should find an efficient way to compute or approximate the denominator of the right hand side of the equation 8.1.

8.3 Statistical Translation Models

Brown et al. [3] proposed five statistical translation models. Regarding previous lecture, you are familiar with model 3. Models 1 and 2 are simpler and have fewer number of parameters. They will be used for training the more complicated models. Model 4 is the most successful implemented one among these and model 5 is the most complicated, but not practical.

One can think of these models as generative translation models that with $p(a, f|e)$ generate string f and alignment a from e. Models 1 and 2 first choose a length for French string f, assuming all reasonable lengths to be equally likely. Then, for each position j they decide which f_j word to place and which e_i to align. In models 3, 4, and 5 another concept becomes important which is called fertility. Fertility of the English word e_i, Φ_i, is a random variable representing the number of French words to which e_i is connected in a random alignment. The latter models parametrize fertility for each word, as a probability distribution over natural numbers.

8.4 Model 1

Model 1 is a bag of word translation model. In this model, $p(a, f|e)$ is expressed as

$$p(a, f|e) = \frac{\eta}{(l+1)^m} \prod_{j=1}^{m} t(f_j|e_{a_j}) \tag{8.2}$$

$t(f_j|e_{a_j})$ denotes translation probability of f_j given e_{a_j} and η denotes $p(m|e)$. This model is versy simple and assumes that all different alignments are equally likely. So permuting French words while preserving their alignment, maintain the $p(f|e)$. $p(f|e)$ can be written as

$$p(f|e) = \frac{\eta}{(l+1)^m} \sum_{a} \prod_{j=1}^{m} t(f_j|e_{a_j}) = \frac{\eta}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j|e_i) \tag{8.3}$$

We exchange the sum and product in this equation because instead of summing over all $(l+1)^m$ possible alignment, we can multiply m brackets, each of them determine the decision for alignment of jth French word, f_j:

$$[t(f_j|e_1) + t(f_j|e_2) + \ldots + t(f_j|e_l)]$$

The important part is how to automatically obtain the conditional translation probabilities, t. Using Expectation Maximization (EM) algorithm, these parameters can be estimated iteratively. Starting by initial values of translation probabilities, we can compute new values for $t(w_f|w_e)$.

We want to maximize the conditional probabilities of pairs of French and English sentences appeared in the training data. Each iteration of EM tries to maximize $\sum_{(f,e)} p(f|e)$, while satisfying the constraints of the form $\sum_{w'_f} p(w'_f|w_e) = 1$.

Brown et al. [3] showed that the following formula solves this optimization problem. $\delta(w', w)$ is equal to one when words w' and w are equal, and zero otherwise.

$$t_{new}(w_f|w_e) = \lambda_{w_e} \sum_{(f,e)} p(f|e) \frac{t(w_f|w_e)}{\sum_{i'=0}^{l} t(w_f|e_{i'})} \underbrace{\sum_{j=1}^{m} \delta(f_j, w_f)}_{count\ w_f\ in\ f} \underbrace{\sum_{i=0}^{l} \delta(e_i, w_e)}_{count\ w_e\ in\ e} \quad (8.4)$$

λ_{w_e} is the normalization factor and the right hand side of the equation 8.4 efficiently computes the expected probability of w_f being connected to w_e in a random alignment.

In summary, we can train the model 1 iteratively. At each step, we replace old values of translation probabilities with new ones computed from equation 8.4, and redo this step until we have reached a maximal probability. Moreover, regarding the fact that $p(f|e)$ has a unique local maximum, different choices for initial t are the same for the EM algorithm and leads us to the global maximum. For detailed proof refer to [3].

8.5 Model 2

In model 1, the probability of aligning words f_j and e_i was independent of their positions in string f and e, j and i respectively. In model 1 we assumed all places to be equally likely and used coefficients $\frac{1}{l+1}$. The only contribution

of model 2 is that it introduces parameter $a(i|j,l,m)$ which is the probability of connecting jth word of French sentence of length m to ith word of English sentence of length l.

The alignment parameter, a, slightly changes equations 8.2, 8.3, and 8.4.

$$p(a,f|e) = \eta \prod_{j=1}^{m} t(f_j|e_{a_j}) a(a_j|j,l,m) \tag{8.5}$$

$$p(f|e) = \eta \sum_{a} p(a,f|e) = \eta \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j|e_i) a(i|j,l,m) \tag{8.6}$$

$$t_{new}(w_f|w_e) = \lambda_{w_e} \sum_{(f,e)} p(f|e) \sum_{j=1}^{m} \sum_{i=0}^{l} \frac{t(w_f|w_e) a(i|j,l,m)}{\sum_{i'=0}^{l} t(w_f|e_{i'}) a(i'|j,l,m)} \delta(f_j,w_f) \delta(e_i,w_e) \tag{8.7}$$

Also $a_{new}(i|j,l,m)$ is estimated by its expected probability over all alignments. The reader can compute its formula which is in the same form of equation 8.7.

Clearly, any instance of parameters of model 1 is valid for model 2 where $\forall_{i,j,m} a(i|j,l,m) = \frac{1}{l+1}$. Thus, For training model 2, we can first train model 1 and then, inject its translation probabilities to model 2. Finally, model 2 can manipulate these probabilities while finding appropriate values for alignment parameters using EM algorithm.

Model 2 can be used to find the most probable alignment for a pair of sentences (f,e). We call this alignment, *Viterbi alignment* and show it by $V(f|e)$. For model 2 we have

$$V_2(f|e)_j = \underset{i}{\text{argmax}} \ t(f_j|e_i) a(i|j,l,m) \tag{8.8}$$

8.6 Generative Translation Process

In models 3, 4, and 5, given an English sentence e, we first pick a value for ϕ_i, the fertility of word e_i. Then, according to e and fertility values, we substitute e_i with a sequence of French words. We call the french expression associated with each English word, a *tablet* and represent the tablet associated with e_i by random variable T_i. You should notice that on the French side, T_i does not precede T_{i+1} always and T_i is empty for words with zero fertility. Random variable T is the sequence of tablets which is called a *tableau*.

A tableau is an elementary translation. After obtaining a tableau, we permute its words to produce f. Π is the random variable for this permutation. Since we show jth word of T_i by $T_{i:j}$, we use random variable $\Pi_{i:j}$ for the position of the word $T_{i:j}$ in the final translation f. It is clear that Π is a permutation of order m (size of f).

The joint likelihood for a tableau τ and permutation π is

$$p(\tau, \pi|e) = p(\tau|e)p(\pi|\tau, e) \tag{8.9}$$

$$\begin{aligned} p(\tau|e) = \; & \prod_{i=1}^{l} p(\phi_i|\phi_1^{i-1}, e) \times p(\phi_0|\phi_1^{l}, e) \times \\ & \prod_{i=0}^{l} \prod_{j=1}^{\phi_i} p(\tau_{i:j}|\tau_{i:1}^{i:j-1}, \tau_0^{i-1}, \phi, e) \end{aligned} \tag{8.10}$$

$$\begin{aligned} p(\pi|\tau, e) = \; & \prod_{i=1}^{l} \prod_{j=1}^{\phi_i} p(\pi_{i:j}|\pi_{i:1}^{i:j-1}, \pi_1^{i-1}, \tau, e) \times \\ & \prod_{j=1}^{\phi_0} p(\pi_{0:j}|\pi_{0:1}^{0:j-1}, \pi_1^{l}, \tau, e) \end{aligned} \tag{8.11}$$

Where x_i^j denotes the sequence $x_i, \ldots, x_j$. It is clear that fertility values are encoded in the tableau. So, we didn't sum over different fertility values to obtain $p(\tau|e)$.

Knowing τ and π, we can determine f and a. However, there are $\prod_{i=0}^{l} \phi_i!$ different pairs of (τ, π) that lead to the same (f, a). The $\phi_i!$ terms are different permutations of words within tablets. We can adapt π such that for different permuations of words within each tablet, we get the same f. Finally, for computing $p(a, f|e)$, we should sum over probability of all pairs of (τ, π) that lead to (a, f).

To make it clear, we give an example here. Consider the string "`x y`" as an English sentence. The nature of above equations suggests a step by step actions to get a final string f. First, we should generate ϕ_1, the fertility of word `x`, then according to that ϕ_2 and finally ϕ_0, the fertility of word `null`. assume

$$\phi = (2, 0, 2)$$

Then, we decide about tablets. T_1 is empty while each of others have two words. τ is a tableau given here:

$$\tau = [\mathtt{n}'_1,\ \mathtt{n}'_2]\ [\ ]\ [\mathtt{y}'_1,\ \mathtt{y}'_2]$$

At the end, we should find the appropriate permutation π. permutation

$$\pi = (1, 3, 2, 4)$$

Table 8.1. Model 3 parameters

	name	description
n	fertility	$n(k\|e_i)$ is the probability of tablet T_i having length k
t	translation	$t(f_j\|e_i)$ is the probability of e_i be translated to f_j, similar model 1 and 2
d	distortion	$d(j\|i,m,l)$ is the probability of f_j be e_i's translation given m and l [1]
p_1	fertility probability of e_0	fertility of e_0 is treated differently using this parameter
p_0		$1-p_1$

leads to French string "$\mathtt{n}'_1\ \mathtt{y}'_1\ \mathtt{n}'_2\ \mathtt{y}'_2$" and alignment vector $(0,2,0,2)$. It's easy to check that

$$\tau' = \mathtt{[n}'_2\mathtt{,\ n}'_1\mathtt{]\ [\ ]\ [y}'_2\mathtt{,\ y}'_1\mathtt{]} \qquad \pi' = (3,1,4,2)$$

lead to the same f and a. Here there are 2!2! pairs of (τ,π) that give us the same a and f. This generative process is the intuition behind equations 8.9, 8.10, and 8.11.

8.7 Model 3

Model 3 was described in the previous lecture. We summarized its parameters in table 8.1. $p(f|e)$ in this model is expressed as

$$p(f|e) = \sum_a \underbrace{\binom{m-\phi_0}{\phi_0} p_1^{\phi_0} p_0^{m-2\phi_0} \prod_{i=1}^{l} n(\phi_i|e_i)\phi_i! \prod_{j=1}^{m} t(f_j|e_{a_j}) d(j|a_j,m,l)}_{p(a,f|e)} \tag{8.12}$$

Model 3 is a kind of generative process described in the previous section. It simplifies parameters of the generative process and equation 8.12 is simplified and unified version of equations 8.9, 8.10, and 8.11. The term $\prod_{i=1}^{l}\phi_i!$ is added to convert $p(\pi,\tau|e)$ to $p(a,f|e)$ as described in section 8.6. However, we don't need $\phi_0!$ because it is canceled out.

Now the problem is how to compute $p(f|e)$ efficiently. Unlike models 1 and 2, we cannot exchange sum and product due to $n(\phi_i|e_i)\phi_i!$ terms. It's a severe problem because for re-estimation step of EM, we have to sum over

all alignments and compute the expected value of each parameter. Also we need $p(f|e)$ in our estimation step for each of training pairs.

Brown et al. [3] approximated sum over all alignments by summing over a portion of more probable ones, ignoring the others. They defined proximity over alignment space by two operations: move and swap. Two alignments a and a' differ by a move, if and only if there is only one index j such that $a_j \neq a'_j$. Similarly, two alignments a and a' differ by a swap if and only if they are equal except at two indexes j and k that $a_j = a'_k$ and $a_k = a'_j$. E.g. assume following alignments

$$a = \texttt{[1 2 1 0 3]} \qquad a' = \texttt{[2 1 1 0 3]} \qquad a'' = \texttt{[2 1 1 1 3]}$$

we can obtain alignment a' from a by a swap and alignments a' and a'' differ by a move. But we cannot derive a'' form a by a swap nor a move. We say two alignments are neighbors if those are the same or differ by a swap or a move. $\mathcal{N}(a)$ denotes the set neighbors of a.

We want to find set S of probable alignments to replace all $\sum_a$ with $\sum_{a \in S}$. $\mathcal{N}(V_3(f|e))$, neighbors of viterbi alignment, seems to be a good candidate for S. However, we cannot efficiently compute $V_3(f|e)$. So we should approximate $V_3(f|e)$ itself.

$b(a)$ is the neighbor of a with greatest $p(b(a)|f, e)$. Recursively computing $b(a)$, $b(b(a))$, etc. we converge to $b^\infty(a)$. We can use $b^\infty(V_2(f|e))$ instead of $V_3(f|e)$. Let $b_{j \mapsto i}(a)$ be the most probable alignment within neighbors of a which preserves f_j and e_i connectivity. $b^\infty_{j \mapsto i}(a)$ is defined similar to $b^\infty(a)$ and $V_{2:j \mapsto i}(f|e)$ is the viterbi alignment that associate f_j with e_i. Brown et al. define S as:

$$S = \bigcup_{i,j} \mathcal{N}(b^\infty_{j \mapsto i}(V_{2:j \mapsto i}(f|e))) \bigcup \mathcal{N}(b^\infty(V_2(f|e))) \tag{8.13}$$

On the other hand, the property of swap and move operations let us to compute $p(a'|f, e)$ when $a' \in \mathcal{N}(a)$ directly from $p(a|f, e)$. E.g. when a' is obtained by move of j from i to i' assuming $i \neq 0$ and $i' \neq 0$

$$\frac{p(a'|f,e)}{p(a|f,e)} = \frac{\phi_{i'}+1}{\phi_i} \frac{n(\phi_{i'}+1|e_{i'})}{n(\phi_{i'}|e_{i'})} \frac{n(\phi_i - 1|e_i)}{n(\phi_i|e_i)} \frac{t(f_j|e_{i'})}{t(f_j|e_i)} \frac{d(j|i,m,l)}{d(j|i',m,l)} \tag{8.14}$$

A similar equation can be written for swap and the case that $i = 0$ or $i' = 0$.

Training model 3, can be done using EM algorithm and re-estimation over set S instead of all alignments. Model 3 has many parameters and strting

from random initial state, doesn't lead us to a good state. So we use model 2 to train model 3. We don't inject parameters from model 2 to 3, but we use $p(a|f,e)$ of model 2 to re-estimate parameters of model 3.

Model 3 is *deficient*. It means that $\sum_{f'} p(f'|e) \neq 1$. The main source of deficiency is its distortion probability. We didn't take into account the positions of f that are already occupied. Thus, we might assign some probability, $d(j|i,m,l)$ to pre-occupied positions. However, the simplicity of model 3 is because of this naive assumption.

8.8 Model 4

In model 4, the distortion parameter of model 3 is adapted to consider the positional relationship between French words within tablets.

In model 3 and 4, we have some English words with fertility 0. We call each word with non-zero fertility a cept and show position of ith cept by $[i]$. So $e_{[i]}$ is the ith cept. For cept $[i]$, we define two parameters: $head_i$ and $\odot_i$. $head_i$ denotes the position of leftmost French word in tablet $T_{[i]}$ and $\odot_i$ is the center of tablet $T_{[i]}$, the average of positions of French words which are associated with $e_{[i]}$.

$$\odot_i = \frac{\sum_{k=1}^{\phi_{[i]}} \pi_{[i]:k}}{\phi_{[i]}}$$

In this model, $d(j|i,m,l)$ is replaced by two parameters: $d_1(x - \odot_{i-1}|e_{[i-1]}, f_j)$ for placing the $head_i$ and $d_{>1}(x - \pi_{[i]:k-1}|f_j)$ for placing $T_{[i]:k}$, the kth word of tablet $T_{[i]}$. Thus, $d_{>1}(-1|\text{the})$ is a value close to one, because "the" is likely to appear at the beginning of $T_{[i]}$ phrase. $d_1(-1|\text{book,blue})$ seems to be much greater than $d_1(1|\text{book,blue})$ because "blue" is more likely to appear before "book".

Knowing parameters d_1, $d_{>1}$, n, and t we can compute probability of an alignment. Starting from $T_{[1]:1}$, first word of tablet associated with the first cept, and then moving to next words of $T_{[1]}$, and then to the next tablets, we can compute distortion probabilities of all words with respect to previous words and their positions. t and n affect the $p(a|f,e)$ in the same way as model 3. For training model 4 we can employ the same trick again; summing over a portion of alignments. Now, the reader can rewrite equations 8.12 and 8.13 for model 4.

Table 8.2. Summery of statistical models

model	description	comment	deficient
1	Lexical translation model		no
2	Absolute reordering of French words		no
3	Fertility is added for each English word	uses generative translation process	yes
4	Relative reordering of French words	most practical	yes
5	Fixes deficiency	not practical	no

Parameters $d_1(x - \odot_{i-1}|e_{[i-1]}, f_j)$ and $d_{>1}(x - \pi_{[i]:k-1}|f_j)$ suffer from sparse data, since there are so many $(e_{[i-1]}, f_j)$ pairs and the vocabulary is huge itself. We come up with this issue by using word classes instead of words. We use $d_1(x - \odot_{i-1}|\mathcal{A}(e_{[i-1]}), \mathcal{B}(f_j))$ and $d_{>1}(x - \pi_{[i]:k-1}|\mathcal{B}(f_j))$ instead where $\mathcal{A}(w_e)$ and $\mathcal{B}(w_f)$ are clustering functions for English and French words respectively. A statistical word clustering algorithm is described in [2].

Model 4 improves model 3, but it is still deficient. In model 4, words can be placed in the same position or before the first or after the last position, in the French string.

8.9 Model 5

In this model we change the distortion probability to take into account all information about vacant positions. So the vacant positions are given to d_1 and $d_{>1}$. This model suffers from sparse data so much, because there are many different permutations for vacant positions.

8.10 Summary of Statistical Models

We summarize all five statistical translation models in table 8.2. These models evolve as their number increase and so bootstrapping technique is used for training.

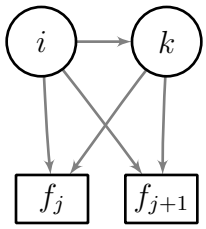

Figure 8.1. HMM for alignment

8.11 HMM-based Alignment

HMM can be also used for solving the problem of word-by-word alignment of strings e and f. Figure 8.1 illustrates a HMM-based alignment model. French words are observations while word positions of english sentence are hidden variables. Each path in this model represents an alignment vetor a where a_j is a hidden node that emits observation f_j. The emission probability, $p(f_j|i)$ is equal to $t(f_j|e_i)$ and transition probability $p(k|i)$ is equal to $p(k|i,l)$ where l is the length of English sentence. The intuition behind this model is that in an alignment a, a_j has strong dependency on a_{j-1} and l and most of the time a_j and a_{j-1} differs by less that 3.

$$p(a, f|e) = \prod_{j=1}^{m} p(a_j|a_{j-1}, l)t(f_j|e_{a_j}) \tag{8.15}$$

Running viterbi algorithm on this HMM, it finds a sequence of hidden variables, a^*, that produces string f; $f_1 f_2 \dots f_m$. a^* is an alignment that maximize $p(a, f|e)$ in terms of equation 8.15. As $p(a|f, e) = \frac{p(a,f|e)}{p(f|e)}$, we can infer that a^* is the most probable alignment for pair of e and f sentences.

This model is similar to model 2. However, In this model, we conditioned a_j on a_{j-1} and l, while in model 2 we conditioned a_j on j, m, and l. Like model 2, in HMM-based alignment model, we can find best alignment and $p(f|e)$ in efficient time because of the algorithms that are available for HMM.

Bibliography

[1] Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, Paul S. Roossin. 1990. *A Statistical Approach to Machine Translation, Computational Linguistics,* 16(2):79-85.

[2] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, Jenifer C. Lai. 1992. *Class-based n-gram models of natural language, Computational Linguistics*, 18(4):467-479.

[3] Peter F. Brown, Stephen A. Della Pietra, Vincent J.Della Pietra, Robert. L. Mercer. 1993. *The mathematics of statistical machine translation: parameter estimation, Computational Linguistics*, 19(2):263-311.

[4] Stephan Vogel and Hermann Ney and Christoph Tillmann. 1996. *HMM-based word alignment in statistical translation, Proceedings of the 16th conference on Computational linguistics*, 836–841.