Word Alignment for Statistical Machine Translation Using Hidden Markov Models

by

Anahita Mansouri Bigvand

A Depth Report Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy

in the
Department of Computing Science

Abstract

Statistical machine translation (SMT) relies on large parallel data between source and target languages. Word alignment is a crucial early step in training of most SMT systems. The objective of the word alignment task is to discover the word-to-word correspondences in a sentence pair. The classic IBM Models 1-5 and the Hidden Markov Model (HMM) have underpinned the majority of the SMT systems to date.

HMMs have been applied to numerous problems in NLP. The key attraction of HMMs is the existence of well-known tractable algorithms for EM parameter estimation (Baum-Welch algorithm) and maximization (Viterbi algorithm). HMMs have been exploited for the word alignment problem. The performance of an improved HMM word alignment model is comparable to that of IBM Model 4 which is arguably the most widely used model for word alignment. Compared to IBM Model 4, HMM is much easier to implement and modify and is more time-efficient to train. This report is a summary of the key papers that use the HMM-based word alignment model for SMT.

Keywords: Statistical machine translation; word alignment; HMM

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Word alignment is an important component of a complete statistical machine translation (SMT) pipeline. The objective of the word alignment task is to discover the word-to-word correspondences in a sentence pair.

Models for word alignment depend on the way they decompose this problem. The classic IBM Models 1-5 (Brown et al., 1993) and the HMM model (Vogel et al., 1996) have underpinned the majority of the SMT systems to date. HMMs have been applied to numerous problems in NLP, such as part-of-speech tagging. The key attraction of HMMs is the existence of well-known tractable algorithms for EM parameter estimation (Baum, 1972) and maximization (Viterbi, 1967). HMMs have been widely studied for the word alignment problem (Och and Ney, 2000a; Toutanova et al., 2002).

The HMM-based word alignment model has been shown to significantly outperform IBM Models 1, 2, and 3 (Och and Ney, 2000a, 2003). IBM 4, 5 and the Och and Ney (2003) variant called IBM Model 6 all outperform IBM Model 3. However, only IBM Model 4, which is a probabilistically deficient version of IBM Model 5, is computationally tractable. HMMs are not deficient (no missing probability mass) and are computationally tractable. On the other hand, IBM model 4 alignment as produced by GIZA++ (Och and Ney, 2000b) are often the best that can be obtained for large parallel corpora. Despite its modelling power and widespread use, IBM model 4 has its own limitations. The parameter estimation of this model is implemented by approximate hill-climbing methods and hence can be very slow, memory-intensive and difficult to parallelize. The performance of a refined HMM model is comparable to that of IBM Model 4, and it is also much easier to understand. Practically, we can train the HMM model by the Forward-Backward algorithm, and by parallelizing the parameter estimation, we can control memory usage, reduce the time needed for training, and increase the corpus used for training. For all these reasons, the focus of this report is on the HMM-based word alignment model and the extensions to this model.

In this report, we will first give a brief overview of word alignment. In chapter 2, we thoroughly present the original HMM-based word alignment model. We categorize the extensions to this model into six types of approaches corresponding to the enhancement they offer: refined alignment (transition) models (section 3.1), NULL-enhanced models (section 3.2), fertility-enhanced models (section 3.3), part-of-speech-enhanced model (section 3.4), word-to-phrase model (3.5) and feature-enhanced model (section 3.6). We will compare their experimental results in chapter 4.

1.1 Word Alignment

In statistical machine translation, the goal is to translate from a source language F into a target language E. Throughout this report, we assume that the source language is French and the target language is English. We have a French sentence $\boldsymbol{f} = f_1^J = f_1, f_2, \ldots, f_J$, and we want to translate it into an English sentence $\boldsymbol{e} = e_1^I = e_1, e_2, \ldots, e_I$. Among all English sentences, we are looking for the one which has the highest probability $Pr(\boldsymbol{e}|\boldsymbol{f})$. Using Bayes rule, we can write:

$$Pr(\boldsymbol{e}|\boldsymbol{f}) = \frac{Pr(\boldsymbol{f}|\boldsymbol{e})Pr(\boldsymbol{e})}{Pr(\boldsymbol{f})} \tag{1.1}$$

As the denominator is independent of $\boldsymbol{e}$, finding the most probable translation $\boldsymbol{e}^*$ will lead to the noisy channel model for statistical machine translation:

$$\begin{aligned} \boldsymbol{e}^* &= \arg\max_{\boldsymbol{e}} Pr(\boldsymbol{e}|\boldsymbol{f}) && (1.2) \\ &= \arg\max_{e} Pr(\boldsymbol{f}|\boldsymbol{e})Pr(\boldsymbol{e}) && (1.3) \end{aligned}$$

where $Pr(\boldsymbol{e})$ is called the language model, while $Pr(\boldsymbol{f}|\boldsymbol{e})$ is called the translation model. The score for a potential translation $\boldsymbol{e}$ is the product of two scores: the language model score which gives a prior probability of the sentence in English, and the translation model score, which indicates the likelihood of seeing the French sentence $\boldsymbol{f}$ as the translation of English sentence $\boldsymbol{e}$. The advantage of using the noisy-channel model is that it allows us to benefit from the language model which is helpful in improving the fluency or grammaticality of the translation. In this report, we will present approaches towards introducing structures into the probabilistic dependencies in order to define translation model $Pr(\boldsymbol{f}|\boldsymbol{e})$ and estimating its parameters from the training data.

We now focus on the problem of modelling the translation probability. It is very hard to model $Pr(\boldsymbol{f}|\boldsymbol{e})$ directly. A key idea in IBM Models (Brown et al., 1993) was to introduce alignment variables to the problem. An alignment $\boldsymbol{a} = a_1, \ldots, a_J$ is a vector of alignment variables where each a_j can take any value in the set $\{1, 2, \ldots, I\}$. The alignment vector

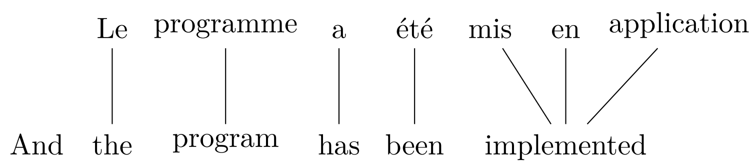


Figure 1.1: An alignment between a French sentence and its translation in English.

specifies the mapping for each French word to a word in the English sentence. Therefore, $a_j = i$ specifies that f_j is aligned to e_i.

Figure 1.1 shows an alignment between a pair of French and English sentences. In this example, $J = 7$ and $I = 6$. Since the length of the French sentence is 7, we have alignment variables $a_1, a_2, \ldots, a_7$ and $a_1, a_2, \ldots, a_7 = \langle 2, 3, 4, 5, 6, 6, 6 \rangle$ specifies the alignment depicted in the figure.

Note that the alignment is many-to-one; i.e. more than one French word can be aligned to an English word while each French word can be aligned to exactly one English word. The fertility ϕ_i of an English word e_i at position i is defined as the number of aligned French words:

$$\phi_i = \sum_{j=1}^{J} \delta(a_j, i) \tag{1.4}$$

where δ is the Kronecker delta function.

Models describing these alignments are called alignment models. The seminal work on word alignment is based on IBM Models 1-5 (Brown et al., 1993). We will briefly discuss IBM Models 1 and 2 in the next section. The focus of this report is on HMM-based word alignment models.

We follow the notational convention of Vogel et al. (1996); we use the symbol $Pr(.)$ to denote general probability distributions without any specific assumption, whereas $p(.)$ is used for model parameters, also known as model-based probability distributions.

1.2 Statistical Alignment Models

In SMT, we try to model the translation probability $Pr(\boldsymbol{f}|\boldsymbol{e})$. Having introduced alignment, rather than modelling $Pr(\boldsymbol{f}|\boldsymbol{e})$, we can try to model $Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e})$ which is called alignment model. We can then calculate the translation model by summing over all possible alignments:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) = \sum_{\boldsymbol{a}} Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) \tag{1.5}$$

Our statistical model depends on a set of parameters θ that can be learned from training data. We use the following notation to show the dependence of the model on the parameters:

$$Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) = Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}, \theta) \tag{1.6}$$

We assume that we have a source of bilingual training data $\boldsymbol{D} = (\boldsymbol{f}^{(k)}, \boldsymbol{e}^{(k)})$ for $k = 1, ..., n$ where $\boldsymbol{f}^{(k)}$ is the k'th French sentence and $\boldsymbol{e}^{(k)}$ is the k'th English sentence, and $\boldsymbol{e}^{(k)}$ is a translation of $\boldsymbol{f}^{(k)}$. We can estimate the parameters of our model using these training examples. The parameters θ are computed by maximizing the likelihood on the parallel training corpus:

$$\theta^* = \arg\max_{\theta} \prod_{k=1}^{n} \Big[\sum_{a} Pr(f^{(k)}, a|e^{(k)}) \Big] \tag{1.7}$$

To perform this maximization, we need to have the alignments. But, the alignments are hidden. Typically, EM algorithm (Dempster et al., 1977) is used in such a scenario. In general, the EM algorithm is a common way of inducing latent structures from unlabelled data in an unsupervised manner. For a given sentence pair, there are many possible alignments. We can find the best alignment $\boldsymbol{a}^*$ as follows:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}, \theta) \tag{1.8}$$

The best alignment $\boldsymbol{a}^*$ is called the *Viterbi alignment* of sentence pair $(\boldsymbol{f}, \boldsymbol{e})$. We can then evaluate the quality of this Viterbi alignment by comparing it to a manually produced reference alignment using the measure explained in section 1.3.

To summarize, given a bilingual training data, we estimate the parameters of our model using the EM algorithm. Using the parameters, we find the best alignment for each sentence pair. The output of word alignment is the given bilingual data with the predicted alignments for each sentence pair. In the following sections, we will describe IBM Models 1 and 2 in a formulation similar to the one for HMM-based alignment model.

1.2.1 IBM Model 1

Introducing the alignment variables allows us to model $Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e})$ instead of $Pr(\boldsymbol{f}|\boldsymbol{e})$. The alignment model can be structured without loss of generality as follows:

$$Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) = \prod_{j=1}^{J} Pr(f_j, a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) \tag{1.9}$$

$$= \prod_{j=1}^{J} Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) Pr(f_j|f_1^{j-1}, a_1^{j}, e_1^I) \tag{1.10}$$

In IBM model 1, we assume a uniform probability distribution for $Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I)$, which depends only on the length of the English sentence:

$$Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) = \frac{1}{I+1} \tag{1.11}$$

where we define e_0 to be a special NULL word that is responsible for generating English words without any corresponding words in the French sentence; $a_j = 0$ specifies that word f_j is generated from the NULL word.

Furthermore, we assume that $Pr(f_j|f_1^{j-1}, a_1^j, e_1^I)$ depends only on f_j and e_{a_j}. Hence, we have:

$$Pr(f_j|f_1^{j-1}, a_1^j, e_1^I) = p(f_j|e_{a_j}) \tag{1.12}$$

Putting everything together, we have the following translation model for IBM Model 1:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) = \frac{1}{(I+1)^J} \sum_a \prod_{j=1}^{J} p(f_j|e_{a_j}) \tag{1.13}$$

Parameter estimation and decoding is explained for IBM Model 2 since Model 1 is a special case of Model 2.

1.2.2 IBM Model 2

In IBM Model 2, we make the same assumptions as in Model 1 except that we assume that $Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I)$ depends only on j, a_j, J and I. A set of alignment parameters is defined as follows:

$$Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) = p(a_j|j, J, I) \tag{1.14}$$

Hence, we have the following translation model for IBM Model 2:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) = \sum_a \prod_{j=1}^{J} p(a_j|j, J, I) p(f_j|e_{a_j}) \tag{1.15}$$

Parameter Estimation

As explained in the previous section, the expectation maximization algorithm or EM algorithm (Dempster et al., 1977) is used for partially-observed data. This algorithm initializes the model parameters (typically with uniform distribution). The algorithm iterates over the expectation and maximization steps until convergence. In the expectation step, it applies the model to the data; in the maximization step, it learns the model from data. The output of this algorithm is the set of translation parameters and alignment parameters. Once we estimate these parameters, we can use them to find the most probable alignment, also called the Viterbi alignment, for any given training example as follows:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} Pr(\boldsymbol{a}|\boldsymbol{f}, \boldsymbol{e}) \tag{1.16}$$

For IBM model 2, the solution to equation 1.16 is as follows:

$$a_j{}^* = \arg\max_{i\in\{0,\ldots,I\}} \ (p(i|j, J, I) \times p(f_j|e_i)) \tag{1.17}$$

1.3 Evaluation with AER

The quality of the alignment methods is evaluated by the performance on a test set for which a gold standard has been established by human annotators. The annotators are asked to specify alignments of two kinds: an S (sure) alignment, for alignments that are unambiguous and a P (possible) alignment, for ambiguous alignments. Thus, the reference alignment obtained may contain many-to-one and one-to-many relationships. The quality of an alignment $A = \{(j, a_j)|a_j > 0\}$ is evaluated using a measure called alignment error rate (AER), which is defined as

$$AER = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} \tag{1.18}$$

1.4 Summary

In this chapter, we provided a background about word alignment. We introduced the basic concepts such as the noisy-channel model. As a useful warm-up before we get to the HMM-based word alignment model, we looked at two statistical alignment models (IBM Models 1 and 2) and explained training and decoding for these models. Finally, we explained AER measure for future references.

Chapter 2

Hidden Markov Alignment Model

In this chapter, we present the HMM-based alignment model proposed by Vogel et al. (1996). We begin with the motivation behind this model. The training and decoding algorithms for this model will be explained in the following sections. We will present the forward-backward and the Viterbi algorithm for the word alignment problem and hence we will explain how to compute the expected counts and posterior probabilities for a parallel training data.

2.1 Motivation

Translation of sentences in parallel texts, especially for language pairs from Indo-European languages, shows a strong localization effect. Words close to each other in a sentence in the source language remain close in the translation sentence. Figure 2.1 shows this effect for the language pair German-English. Note that we visualize the word alignment task by a matrix where alignments between words are represented by points in the alignment matrix. Each word in the German sentence is aligned to a word in the English sentence. Alignments tend to preserve their local neighbourhood when going from German to English, as shown in Figure 2.1.

2.2 Alignment with HMM

As explained previously, introducing the alignment variables allows us to model $Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e})$ instead of $Pr(\boldsymbol{f}|\boldsymbol{e})$. The alignment model can be structured without loss of generality as follows:

$$Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) = \prod_{j=1}^{J} Pr(f_j, a_j | f_1^{j-1}, a_1^{j-1}, e_1^I) \quad (2.1)$$

$$= \prod_{j=1}^{J} Pr(a_j | f_1^{j-1}, a_1^{j-1}, e_1^I) Pr(f_j | f_1^{j-1}, a_1^{j}, e_1^I) \quad (2.2)$$

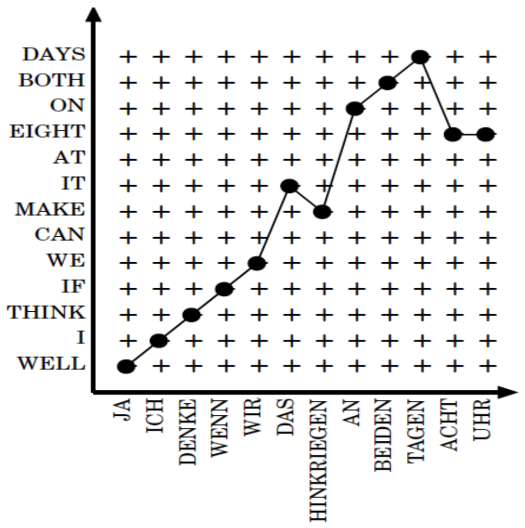


Figure 2.1: Word alignment for a German-English sentence pair (figure from Vogel et al. (1996))

In the Hidden Markov alignment model, we assume a first-order dependence for the alignments a_j; also, the translation probability is assumed to be dependent on the word at position a_j and not on the one at position a_{j-1}. Hence, we have:

$$Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) = p(a_j|a_{j-1}, I) \tag{2.3}$$

$$Pr(f_j|f_1^{j-1}, a_1^{j}, e_1^I) = p(f_j|e_{a_j}) \tag{2.4}$$

Putting everything together, we have the following basic HMM-based model:

$$Pr(f|e) = \sum_a \prod_{j=1}^{J} [\ p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j})\ ] \tag{2.5}$$

with two components: the alignment probability (transition probability) $p(a_j|a_{j-1}, I)$ or $p(i|i', I)$, and the translation probability (emission probability) $p(f_j|e_{a_j})$. Figure 2.2 shows the graphical model of the HMM-based word alignment model.

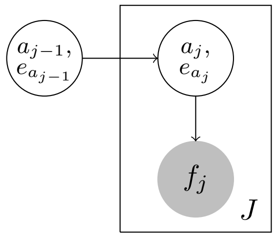


Figure 2.2: Graphical model of the basic HMM-based word alignment model

Estimating the alignment parameters $p(i|i', I)$ using the conventional maximum likelihood (ML) criterion depends on the expectation of consecutive French words generated from the English word at position i' and i with English sentence length I. This clearly could suffer from sparsity within the available bitext. To address this problem, Vogel et al. (1996) make the alignment parameters independent of the absolute word positions. It is assumed that the alignment probabilities $p(i|i', I)$ depend only on the jump width $(i - i')$. Hence, the alignment probabilities are estimated using a set of distortion parameters $c(i - i')$ as follows:

$$p(i|i', I) = \frac{c(i - i')}{\sum_{i''=1}^{I} c(i'' - i')} \tag{2.6}$$

where at each iteration $\{c(i - i')\}$ is the fractional count of transitions with jump width $d = i - i'$:

$$c(d) = \sum_{j=1}^{J-1} \sum_{i=1}^{I} Pr(a_j = i, a_{j+1} = i + d|\boldsymbol{f}, \boldsymbol{e}, \theta) \tag{2.7}$$

where θ is the model parameters obtained from the previous iteration and the terms $Pr(a_j = i, a_{j+1} = i + d|\boldsymbol{f}, \boldsymbol{e}, \theta)$ can be efficiently computed using the Forward-Backward algorithm (Rabiner, 1989).

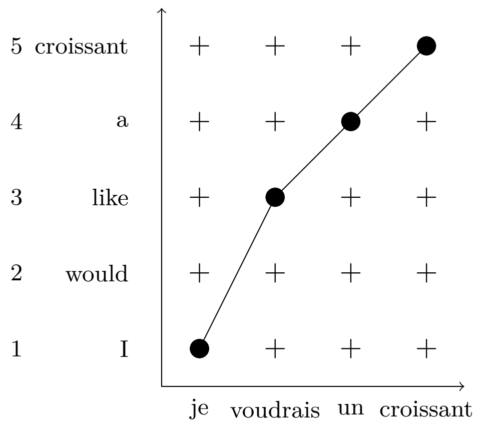
To illustrate how the alignment model is computed for a given sentence pair and its alignment, consider the following example. We visualize the word alignment task by a matrix as in Figure 2.3. Here, alignments between words are represented by points in the alignment matrix. We are given the alignment vector $\boldsymbol{a} = \langle 1, 3, 4, 5 \rangle$. To assign a probability to the alignment model using the HMM model, we consider the first alignment point in the figure, i.e. (je, I).[1] Hence, the emission probability $p(je|I)$ is considered for this point. The second alignment point is (voudrais, like). As the previous French word is aligned to English word I, which is at position 1 in the English sentence, we have a transition to position 3 in the English sentence for the word *like*. Therefore, we have to multiply $p(3|1, 5)$ and also $p(voudrais|like)$ for the emission probability. Similarly, we have a transition probability $p(4|3, 5)$ because of jumping to position 4 for a and an emission probability for $p(un|a)$. Finally, we multiply transition probability $p(5|4, 5)$ and emission probability $p(croissant|croissant)$.

Suppose we are given $c(1) = 2$ and $c(2) = 1$. Computing the transition probabilities are straightforward; for example, $p(3|1, 5) = \frac{c(2)}{c(1)+c(2)} = \frac{1}{3}$.

2.2.1 Training

Parameter estimation of the HMM-based word alignment model can be done using the Baum-Welch (Baum, 1972) algorithm which makes use of the forward-backward algorithm. Forward-backward algorithm is a dynamic programming algorithm that makes it possible to

[1]For the sake of simplicity, we have not included the initial state probabilities in this example.



$$\begin{aligned} p(\boldsymbol{a}, \boldsymbol{f}|\boldsymbol{e}) &= \textstyle\prod_{j=1}^{J} p(a_j|a_{j-1}, I)p(f_j|e_{a_j}) \\ &= p(je|I)\times \\ &\quad p(3|1,5) \times p(voudrais|like)\times \\ &\quad p(4|3,5) \times p(un|a)\times \\ &\quad p(5|4,5) \times p(croissant|croissant) \end{aligned}$$

Figure 2.3: Alignment model computation for a HMM-based model for a French-English sentence pair.

avoid the summation over all alignments. We will look at this algorithm for the HMM-based word alignment model here. A compact representation of the HMM model for alignment is $\theta = \{p(i|i', I), p(f|e), p(i)\}$ where $\{p(i|i', I)\}$ are the transition probabilities, $\{p(f_j|e_i)\}$ are the emission probabilities and $\{p(i)\}$ are the initial state probabilities. Note that we include the initial state probabilities here to give a more precise HMM model. The initial state distribution is defined as $p(i) = Pr(a_1 = i)$ (i.e. $p(i)$ is the probability of aligning the first word in the French sentence to the i-th word in the English sentence). The forward and backward probabilities can be computed efficiently using the recursive definitions, as shown in Figure 2.4.

Posterior Probabilities

With the forward and backward probabilities, we can calculate the posterior probabilities. The probability of aligning the j-th French word to the i-th English word is defined as follows:

$$\gamma_i(j) = Pr(a_j = i|\boldsymbol{f}, \theta) = \frac{\alpha_i(j)\beta_i(j)}{\sum_{l=1}^{I} \alpha_l(J)} \tag{2.8}$$

The probability of aligning the j-th French word to the i-th English word and $(j-1)$-th French word to the i'-th English word is defined as follows:

$$\xi_{i'i}(j-1) = Pr(a_{j-1} = i', a_j = i|\boldsymbol{f}, \theta) = \frac{\alpha_{i'}(j-1)p(i|i', I)\beta_i(j)p(f_j|e_i)}{\sum_{l=1}^{I} \alpha_l(J)} \tag{2.9}$$

Parameter Re-estimation

The Baum-Welch algorithm adjusts the model parameters using the expected couts. Let $\boldsymbol{D}$ denote the parallel data. Let $c(f, e)$ be the posterior count accumulated over all training

Algorithm 1 Forward-backward($\boldsymbol{f}, \boldsymbol{e}, \theta$)

```
-given: A French sentence $f_1 \dots f_J$, an English sentence $e = e_1 \dots e_I$ and the
HMM parameter set $\theta = \{p(i|i', I), p(f_j|e_i), p(i)\}$
Let $\alpha_i(j) = Pr(f_1 \dots f_j, a_j = i)$ be the forward probabilities
Base case:
    $\alpha_i(1) = p(i)p(f_1|e_i)$ for $i \in 1, \dots, I$
Recursive case :
    $\alpha_i(j) = p(f_j|e_i) \sum_{i'=1}^{I} \alpha_{i'}(j-1).p(i|i', I)$ for all $i \in 1, \dots, I$ and $j \in 2, \dots, J$
Let $\beta_i(j) = Pr(f_{j+1} \dots f_J|a_j = i)$ be the backward probabilities
Base case:
    $\beta_{i'}(J) = 1$ for all $i' \in 1, \dots, I$
Recursive case :
    $\beta_{i'}(j-1) = \sum_{i=1}^{I} \beta_i(j).p(i|i', I).p(f_j|e_i)$ for all $i' \in 1, \dots, I$ and $j \in 2, \dots, J$
```

Figure 2.4: The FORWARD-BACKWARD algorithm

sentences of the English word e generating the French word f. Hence, we have

$$c(f, e) = \sum_{(\boldsymbol{f},\boldsymbol{e}) \in \boldsymbol{D}} \sum_{i,j} \gamma_i(j) \delta(f_j, f) \delta(e_i, e) \tag{2.10}$$

The translation probabilities (emission parameters) are then estimated as

$$p(f|e) = \frac{c(f, e)}{\sum_{f'} c(f', e)} \tag{2.11}$$

Let $c(i', i, I)$ be the posterior count of transitions with jump width $(i - i')$ over all training sentences:

$$c(i', i, I) = \sum_{(\boldsymbol{f},\boldsymbol{e}) \in \boldsymbol{D}} \sum_{i',i} c(i - i') \delta(|\boldsymbol{e}|, I) \tag{2.12}$$

where $c(i - i')$ is computed using equation 2.7 and $|\boldsymbol{e}|$ denotes the length of the sentence $\boldsymbol{e}$. Note that the terms in this equation are the transition posterior probabilities ξ. The alignment probabilities (transition parameters) are then estimated as

$$p(i|i', I) = \frac{c(i', i, I)}{\sum_{i''=1}^{I} c(i', i'', I)} \tag{2.13}$$

2.2.2 Decoding

After training, Viterbi decoding is used to find the best alignment $\boldsymbol{a}^*$:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} \prod_{j=1}^{J} [\ p(a_j|a_{j-1}, I).p(f_j|e_{a_j})\ ] \tag{2.14}$$

The Viterbi algorithm is shown in Figure 2.5. Both the Viterbi and the forward-backward algorithms can be computed in $O(I^2 J)$.

Algorithm 2 Viterbi($\boldsymbol{f}, \boldsymbol{e}, \theta$)

```
-given: A French sentence $f_1 \dots f_J$, an English sentence $e = e_1 \dots e_I$ and the
HMM parameter set $\theta = \{p(i|i', I), p(f_j|e_i), p(i)\}$
Base case:
    $V[i, 1] = p(i)p(f_1|e_i)$   for $i \in 1, \dots, I$
Recursive case :
    $V[i, j] = \max_{i'}\{V[i', j-1] \cdot p(i|i', I) \cdot p(f_j|e_i)\}$   for all $i \in 1, \dots, I$ and $j \in 2, \dots, J$
The best score is $\max_i V[i, J]$
To find the most probable alignment, trace back using the $V$ matrix
```

Figure 2.5: The Viterbi algorithm

To avoid the summation over all possible alignments in equation 2.5, Vogel et al. (1996), use the maximum approximation where only the Viterbi alignment is used to collect the counts:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) \cong \max_{\boldsymbol{a}} \prod_{j=1}^{J} [\ p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j})\ ] \tag{2.15}$$

Later on, Och and Ney (2000a) use the Baum-Welch algorithm to train the parameters of the model efficiently.

2.3 Summary

In this chapter, we presented the original HMM-based word alignment model. After presenting the motivation behind the model, we showed how to train and decode the HMM model using the Baum-Welch and Viterbi algorithms. We gave a detailed explanation of forward-backward algorithm for the word alignment problem. The formulations for the expected counts and posterior probabilities were also provided. Also, we provided an example to illustrate how the alignment model can be computed using an HMM model. In the next chapter, we will discuss the methods that extend the basic HMM model.

Chapter 3

Extensions and Improvements

In this chapter, we will present some of the extensions to the HMM alignment model. We first look at a group of methods that propose refined alignment models in section 3.1. Then, we give an overview of the methods that address empty word problem for the HMM model in section 3.2. We discuss methods that model fertility in an HMM-based model in section 3.3, and a method that incorporates part of speech tag information in the translation model in section 3.4. We also explore a word-to-phrase HMM alignment model in section 3.5. A feature-enhanced HMM model is presented in section 3.6.

3.1 Refined Alignment Models

The motivation behind these methods is that the transition model in the HMM-based alignment model is coarse. Transition probabilities only depend on the jump width from the last state to the next state. They, therefore, enhance the transition model in the HMM. We are going to look at three methods : Och and Ney (2000a), Och and Ney (2003) and He (2007).

3.1.1 Class Dependent Transition Models

Och and Ney (2000a) focuses on improving the transition probability. The motivation behind their approach is that the count table $c(i-i')$ has only $2 \cdot I_{max} - 1$ entries which is suitable for a small corpus, but for a large one, it is possible to give an improved model for $Pr(a_j|f_1^{j-1}, a_1^{j-1}, I)$. For instance, the effect of dependence on the surrounding words such as $e_{a_{j-1}}$ or f_j is analyzed. As conditioning on all English words (or French words) would result in a huge number of parameters, equivalence classes G over the English and French words are used. G is a mapping of words to classes. The categorization of words into classes (here, 50 classes) is performed using the statistical learning procedure described in Kneser and Ney (1993). Och and Ney (2000a) extend the transition probabilities to be

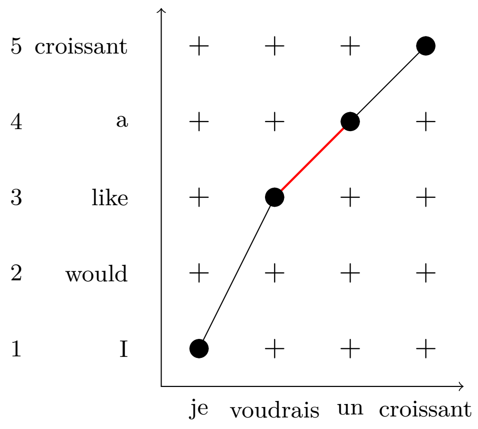


Figure 3.1: Word-dependent transition probability $p(4|3, like, 5)$ for the red edge.

class dependent as follows:

$$p(a_j - a_{j-1}|G(e_{a_j}), G(f_j), I) \tag{3.1}$$

Och and Ney (2003) extend the alignment parameters to include a dependence on the class of the preceding English word:

$$p(a_j|a_{j-1}, G(e_{a_{j-1}}), I) \tag{3.2}$$

3.1.2 Word-Dependent Transition Model

Knowledge of transition probabilities given a particular English word e is not modelled in the original HMM model. To put it more simply, knowledge of jumping from e to another position, e.g., jumping forward (monotonic alignment) or backward (non-monotonic alignment) is not modelled. To improve the transition model, He (2007) extends the transition probabilities to be word-dependent. Therefore, the word-dependent HMM model is as follows:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) = \sum_{a} \prod_{j=1}^{J} [\ p(a_j|a_{j-1}, e_{a_{j-1}}, I).p(f_j|e_{a_j})\ ] \tag{3.3}$$

For example, the transition probability of the red edge in Figure 3.1 is $p(4|3, like, 5)$.

We need to estimate the transition parameter $p(a_j|a_{j-1}, e_{a_{j-1}}, I)$. Consequently, the full parameter set that we need to estimate is $\theta = \{p(i|i', e_{i'}, I), p(f_j|e_i)\}$. The estimation for word-dependent transition probabilities $p(i|i', e_{i'}, I)$ can be carried out using maximum likelihood training:

$$p(i|i', e, I) = \frac{c(i - i'; e)}{\sum_{i''=1}^{I} c(i'' - i'; e)} \tag{3.4}$$

where at each iteration the word-dependent distortion set $\{c(i-i';e)\}$ is computed as follows:

$$c(d;e) = \sum_{j=1}^{J-1} \sum_{i=1}^{I} \delta(e_{a_j}, e) Pr(a_j = i, a_{j+1} = i + d | \boldsymbol{f}, \boldsymbol{e}, \theta) \tag{3.5}$$

where $d = i - i'$ is the jump width and δ is the Kronecker delta function. For non-frequent words, the data samples for $c(d;e)$ is very limited and hence leads to data sparsity problem. To address this problem, maximum a posteriori (MAP) framework is applied (Gauvain and Lee, 1994) where an appropriate prior $g(\theta|\boldsymbol{e})$ is used to incorporate prior knowledge into the model parameter estimation:

$$\theta_{MAP} = \arg\max_{\theta} p(\boldsymbol{f}|\boldsymbol{e}, \theta) g(\theta|\boldsymbol{e}) \tag{3.6}$$

The relation between ML and MAP estimation is through the Bayes theorem:

$$p(\theta|\boldsymbol{f}, \boldsymbol{e}) \propto p(\boldsymbol{f}|\boldsymbol{e}, \theta) g(\theta|\boldsymbol{e}) \tag{3.7}$$

As the transition model $p(i|i', e_{i'}, I)$ is a multinomial distribution, its conjugate prior is a Dirichlet distribution with the form

$$g(p(i|i', e_{i'}, I)|\boldsymbol{e}) \propto \prod_{i=1}^{I} p(i|i', e_{i'}, I)^{v_{i',i}-1} \tag{3.8}$$

where $v_{i',i}$ is the set of hyper-parameters of the prior distribution. By substituting 3.8 in 3.7, the iterative MAP training formula for transition model is obtained as

$$p_{MAP}(i|i', e, I) = \frac{c(i-i';e) + v_{i',i} - 1}{\sum_{i''=1}^{I} c(i''-i';e) + \sum_{i''=1}^{I} v_{i',i''} - I} \tag{3.9}$$

Hyper-parameter set $\{v_{i',i}\}$ of the prior distribution is set to word-independent transition probabilities:

$$v_{i',i} = \tau . p(i|i', I) + 1 \tag{3.10}$$

where τ is a positive parameter which needs to be tuned on a held-out dataset. Substituting 3.10 in 3.9, we obtain the MAP-based transition model:

$$p_{MAP}(i|i', e, I) = \frac{c(i-i';e) + \tau . p(i|i', I)}{\sum_{i''=1}^{I} c(i''-i';e) + \tau} \tag{3.11}$$

In this new formulation, for frequent words with a substantial amount of data samples for $c(d;e)$, the summation in the denominator is large; therefore, $p_{MAP}(i|i', e, I)$ is dominated by the data distribution. On the contrary, for rare words with low counts of $c(d;e)$, $p_{MAP}(i|i', e, I)$ will approach to the word-independent model. Note that we can vary τ to control the contribution of the prior in this model. For instance, for a small τ, a weak prior

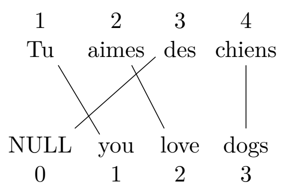


Figure 3.2: Introducing a NULL word at position 0 as Brown et al. (1993) to generate the French word *des* with no corresponding translation in the English sentence.

is applied, and the transition probability is more dependent on the training data of that word. Conversely, for a large τ, a stronger prior is applied, and the model will approach to the word-independent model.

3.2 Empty Word

Some words in a French sentence may have no relation to any of the words in the corresponding English sentence. To model this in the IBM word alignment models, Brown et al. (1993) define e_0 to be a special NULL (empty) word that is treated just like another English word; hence, $a_j = 0$ specifies that f_j is generated from a NULL word. The inclusion of a NULL word is helpful since we still want to align each French word to an English word. If we do not model NULL, we have to align those French words with no correspondences in the English sentence, to arbitrary unrelated words in the English sentence. This results in a model with a high alignment error rate.

Figure 3.2 shows an example of a pair of sentences where the French word *des* has no corresponding word in the English sentence. IBM models align *des* to NULL.

In the original formulation of the HMM model, there is no empty word that is responsible for generating French words with no corresponding English words. A direct inclusion of the NULL word in the HMM model by adding an e_0 as in Brown et al. (1993) is problematic if we want to model the jump distances $i - i'$, as the position $i = 0$ for the NULL word is chosen arbitrarily. W explain two methods thet address this issue: Och and Ney (2000a) and Toutanova et al. (2002).

3.2.1 Adding a Group of NULLs

Och and Ney (2000a) extend the HMM network by I NULL words e_{I+1}^{2I} such that the English word e_i has a corresponding NULL word e_{i+I}. The NULL position chosen by the model is determined by the previously visited English word. The following constraints are enforced for the transition probabilities in the HMM network ($i \leq I, i' \leq I$):

$$p(i+I|i',I) = p_0.\delta(i,i') \tag{3.12}$$

$$p(i+I|i'+I,I) = p_0.\delta(i,i') \tag{3.13}$$

$$p(i|i'+I,I) = p(i|i',I) \tag{3.14}$$

The parameter p_0 is the probability of transitioning to the NULL word, which has to be optimized on a held-out dataset. Och and Ney (2000a) set $p_0 = 0.2$ for their experiments. To illustrate how the NULL word is chosen in this approach, we use the following example. Consider the sentence pair given previously. As the length of the English sentence is 3, we extend the HMM network by 3 NULL words as in Figure 3.3. These NULL words are added to the end of the English sentence. The French word *des* has no corresponding word in the English sentence; hence, it should be aligned to a NULL. However, there are three possible choices. Since the previous French word *aimes* is aligned to a non-NULL word, we fall into the case 3.12 . The word *aimes* is aligned to *love* which is at position 2. Therefore, the position of the NULL word for *des* is $2 + 3 = 5$, as shown in Figure 3.3.

The purpose of the above constraints is to preserve the locality in the HMM when an alignment to NULL is necessary. Suppose, we use the IBM NULL model for our HMM, as in Figure 3.2. The jump sequence $\{a_j - a_{j-1}\}$ is $\{1, -2, 3\}$ where -2 is for jumping to NULL (from position 2 to position 0) and 3 is for jumping back to position 3 from NULL. Note that transition probabilities for jump widths of -2 and 3 are caused by the inclusion of empty word at position 0, and these probabilities should not be included in the alignment model computation. The method proposed by Och and Ney (2000a) resolves this problem. In this method, the transition probabilities used in the computation of the alignment model are $p(2|1,3)$, $p(5|2,3)$ and $p(3|5,3)$ where $p(5|2,3)$ is p_0 due to constraint 3.12 and $p(3|5,3)$ is $p(3|2,3)$ due to constraint 3.14. The locality is preserved even though we align *des* to NULL because the last English word (at position 2) is remembered by the HMM model so that the next alignment to a non-NULL word is computed using $p(3|2,3)$.

3.2.2 Translation Model for NULL

This method presents a new generative model for the source language words that do not have any correspondences in the target language (Toutanova et al., 2002). The translation probabilities for the French words with no correspondences in English is modelled such that these words are generated from NULL and also from the next word in the French sentence by a mixture model. For instance, in the pair *des chiens* in Figure 3.2, *chiens* contributes extra information in generation of *des*. The new formulation for the probability of a French word given that it does not have a corresponding word in the English sentence is:

$$p(f_j|a_j = 0) = \lambda p(f_j|f_{j+1}, e_{a_j} = \text{NULL}) + (1-\lambda)p(f_j|e_{a_j} = \text{NULL}) \tag{3.15}$$

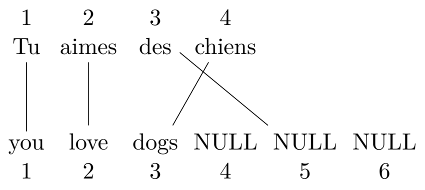


Figure 3.3: Introducing a group of NULL words as Och and Ney (2000a) to generate the French word *des*.

The probabilities $p(f_j|f_{j+1}, e_{a_j} = \text{NULL})$ are re-estimated from the training corpus using the EM algorithm. Note that the dependence of a French word on the next French word requires a change in the generative model. First, alignments are proposed for all words in the French sentence and then French words are generated given their alignment starting from the end of the sentence towards the beginning. For this model, there is an efficient dynamic programming algorithm similar to the forward-backward algorithm that can be used for computations in EM.

3.3 Modelling Fertility

One major advantage of IBM models 3-5 over the HMM model is the presence of a model for English word fertility. Thus, knowledge that some French words translates as phrases in English was incorporated in these models. HMM-based word alignment model has no memory, but the previous alignment, about how many words have been aligned to an English word, and this memory is not used to decide whether to generate more words from this word. This decision is independent of the word and is estimated over all the words in the corpus as estimating the transition probability with a jump of size 0 is computed using equation 2.6. Toutanova et al. (2002) extend the HMM model such that deciding whether to generate more French words from the previous English word $e_{a_{j-1}}$ or to move to another English word depends on the word $e_{a_{j-1}}$. To do this, they introduce a factor $p(stay|e_{a_{j-1}})$, where the boolean random variable *stay* depends on the English word $e_{a_{j-1}}$. The rationale for this method is that as in most cases words with fertility more than one generate consecutive words in the source language, this method approximately models fertility. They change the baseline model, equation 2.5, as follows:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) = \sum_{\boldsymbol{a}} \prod_{j=1}^{J} [\tilde{p}(a_j|a_{j-1}, e_{a_{j-1}}, I) p(f_j|e_{a_j})] \tag{3.16}$$

where

$$\begin{aligned}\tilde{p}(a_j|a_{j-1}, e_{a_{j-1}}, I) = \delta(a_j, a_{j-1})p(stay|e_{a_{j-1}})\\ + (1 - \delta(a_j, a_{j-1}))(1 - p(stay|e_{a_{j-1}}))p(a_j|a_{j-1}, I)\end{aligned}$$

A jump of size zero in the new alignment (transition) probabilities $\tilde{p}(a_j|a_{j-1}, e_{a_{j-1}}, I)$ depends on the English word $e_{a_{j-1}}$.

To handle the sparsity problem in estimating $p(stay|e_{a_{j-1}})$, smoothing is done using a probability of a jump zero as the prior:

$$p(stay|e_{a_{j-1}}) = \lambda p_{ZJ} + (1 - \lambda)p(stay|e_{a_{j-1}}) \tag{3.17}$$

where p_{ZJ} is the alignment probability of the baseline model for a jump of size zero $p_{ZJ} = Pr(a_j = i|a_{j-1} = i, I)$.

Modeling fertility is challenging in the HMM framework as it violates the Markov assumption. Whereas the HMM jump model considers only the prior state, fertility requires looking across the whole state space. Therefore, the standard forward-backward and Viterbi algorithms do not apply.

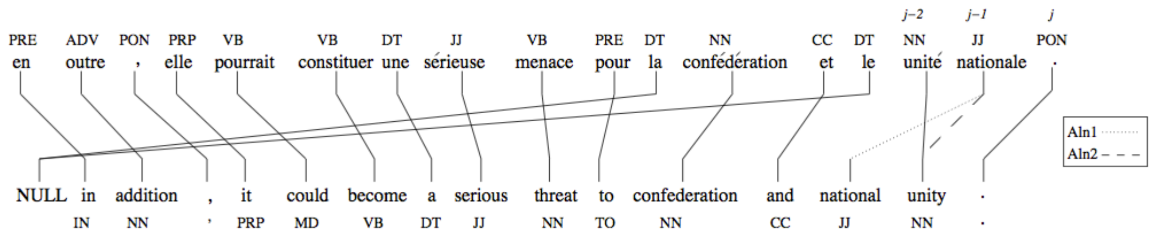
Zhao and Gildea (2010) build a fertility hidden Markov model by adding fertility to the HMM. Their model assumes that fertility ϕ_i for a word e_i follows a Poisson distribution Poisson($\phi_i, \lambda(e_i)$). Compared to IBM Models 3-5, this model has a much smaller number of parameters to learn; one parameter for each English word. In this method, we estimate the parameters using the EM algorithm. However, to compute the expected counts, we have to sum over all possible alignments, which is exponential. Gibbs sampling is thus used to approximate the expected counts.

3.4 Part of Speech Tags in a Translation Model

This method extends the HMM model by incorporating part of speech (POS) tag information of the source and target languages in the translation model (Toutanova et al., 2002). Basic HMM model introduces some alignment irregularities. For instance, the transition of the $NP \rightarrow JJ\ NN$ rule to $NP \rightarrow NN\ JJ$ from English to French[1]. There are two main reasons why translation probabilities may not catch such irregularities in monotonicity. (1) When both English adjective and noun are unknown words, the translation probabilities will be close to each other after smoothing. (2) When the adjective and noun are words that are frequently seen together in English and therefore there will be an indirect association between the English noun and the translation of the English adjective. In such cases, the

[1]Word order changes in translating an adjective-noun pair in English to French.

word translation probabilities are not differentiating enough and alignment probabilities become the dominating factor to make a decision about which English word should be aligned to f_j. Figure 3.4 shows how the basic HMM model makes such an alignment mistake. The table shows the alignment and translation probabilities of two alignments, $Aln1$ and $Aln2$, for the last three words. $Aln1$ correctly aligns the pair of adjective and noun in French to the corresponding pair in English while $Aln2$ incorrectly aligns both French noun and adjective to the English word. (i.e. $e_{a_{j-1}} = unity$ and $e_{a_{j-2}} = unity$). Since the jump width sequences $\{(a_{j-1} - a_{j-2}), (a_j - a_{j-1})\}$ for $Aln2$ is $\{0, 1\}$ which are more probable than $\{-1, 2\}$ for $Aln1$, and the translation probabilities are not differentiating enough, $Aln2$ is preferred by the HMM model.



	e_{a_j}	$e_{a_{j-1}}$	$e_{a_{j-2}}$	P($a_{j-1} \mid a_{j-2}, l$)	P($f_{j-1} \mid e_{a_{j-1}}$)	P($a_j \mid a_{j-1}, l$)	P($f_j \mid e_{a_j}$)	total
Aln1	.	national	unity	0.0292	0.5741	0.05357	0.9766	8.77x10^{-4}
Aln2	.	unity	unity	0.0862	0.2789	0.31	0.9766	7.2x10^{-3}

Figure 3.4: The basic HMM model makes a simple alignment error (figure from Toutanova et al. (2002)).

3.4.1 POS Tags for Translation Probabilities

Let $\boldsymbol{eT}$ and $\boldsymbol{fT}$ be the possible part of speech tag sequences of the sentences $\boldsymbol{e}$ and $\boldsymbol{f}$. The model with part of speech tags for translation probabilities has the following form:

$$Pr(\boldsymbol{f}, \boldsymbol{fT} | \boldsymbol{e}, \boldsymbol{eT}) = \sum_{a} \prod_{j=1}^{J} [\, p(a_j | a_{j-1}, I).p(fT_j | eT_{a_j}).p(f_j | e_{a_j}) \qquad (3.18)$$

This model introduces tag translation probabilities as an extra factor to equation 2.5. This factor boosts the translation probabilities for words of part of speech that are often translations of each other. Hence, it provides a prior knowledge of the translations of a word based on its part of speech. This factor should not be too sharp that dominates the alignment and translation probabilities. To avoid this potential problem, smoothing is done for tag translation probabilities:

$$p(fT_j | eT_{a_j}) = \lambda \tilde{p}(fT_j | eT_{a_j}) + (1 - \lambda) \frac{1}{T} \qquad (3.19)$$

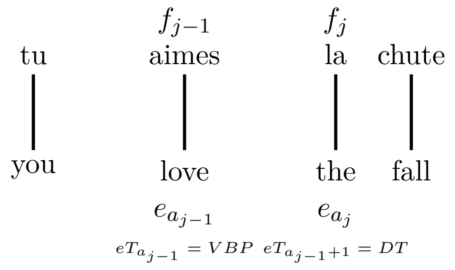


Figure 3.5: Part of speech tag information for modelling local word order variation

where T is the size of the French tag set and $\lambda = 0.1$. It is necessary to set λ to a small value to prevent translation probabilities from becoming very sharp in EM and dominating the alignment and translation probabilities.

In section 3.1.1, we discussed methods that explores conditioning the alignment probabilities on the class of the English word $e_{a_{j-1}}$ and/or that of French word f_j. Toutanova et al. (2002) investigates whether they can improve the model by conditioning on the POS tags of those words or even more words around the alignment position. For instance, consider using $p(a_j|a_{j-1}, eT_{a_{j-1}-1}, eT_{a_{j-1}}, eT_{a_{j-1}+1})$ as the alignment probability. This model is helpful in modelling local word order variations. Figure 3.5 shows an example where the probability of aligning $f_j = la$ (with $fT_j = DT$) to *the* will be boosted knowing that $eT_{a_{j-1}} = VBP$ and $eT_{a_{j-1}+1} = DT$.

3.5 HMM Word and Phrase Alignment

This method develops a generative probabilistic model of Word-to-Phrase (WtoP) alignment (Deng and Byrne, 2008). Suppose, we have a target sentence in English $\boldsymbol{e} = e_1^I$ and its translation in the source language French $\boldsymbol{f} = f_1^J$. Here, the term *phrase* is used to refer to any subsequence in the source sentence. We define the phrase count variable K, which specifies that the French sentence is segmented as a sequence of K phrases: $\boldsymbol{f} = v_1^K$.

Each French phrase is generated as a translation of an English word. The correspondence between English words and French phrases are determined by the alignment sequence a_1^K. The length of each phrase is specified by the random process ϕ_1^K which is constrained to satisfy $\sum_{k=1}^K \phi_k = J$. French phrases are allowed to be generated by NULL. Hence, a binary NULL prediction sequence h_1^K is introduced. If $h_k = 0$, then v_k is aligned to NULL; if $h_k = 1$, then v_k is aligned to e_{a_k}. Taking into account all these quantities, the alignment can be treated as $\boldsymbol{a} = (\phi_1^K, a_1^K, h_1^K, K)$. We can rewrite alignment model $Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e})$ as:

$$\begin{aligned} Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) = Pr(v_1^K, K, a_1^K, h_1^K, \phi_1^K|\boldsymbol{e}) = {} & Pr(K|J, \boldsymbol{e}) \\ & \times Pr(a_1^K, \phi_1^K, h_1^K|K, J, \boldsymbol{e}) \\ & \times Pr(v_1^K|a_1^K, \phi_1^K, h_1^K, K, J, \boldsymbol{e}) \end{aligned} \tag{3.20}$$

We describe the component distributions here:

- Phrase count distribution: $Pr(K|J, \boldsymbol{e})$ specifies the distribution over the number of phrases in the French sentence given the length of the French sentence and the English sentence. A single parameter distribution is used for phrase count distribution:

  $$Pr(K|J, I) \propto \eta^K \tag{3.21}$$

  where $\eta \geq 1$ controls the segmentation of the French sentence into phrases such that larger values of η leads to French sentence segmentation with many short phrases. η is used as a tuning parameter to control the length of phrases.

- Word-to-Phrase alignment: The alignment is modelled as a Markov process that specifies the length of phrases and their alignment with English words:

  $$\begin{aligned} Pr(a_1^K, \phi_1^K, h_1^K|K, J, \boldsymbol{e}) &= \prod_{k=1}^{K} Pr(a_k, \phi_k, h_k|a_{k-1}, \phi_{k-1}, \boldsymbol{e}) \\ &= \prod_{k=1}^{K} p(a_k|a_{k-1}, h_k; I) d(h_k) n(\phi_k; e_{a_k}) \end{aligned} \tag{3.22}$$

  The word-to-phrase alignment a_k is a first order Markov process as in word-to-word HMM (Vogel et al., 1996). It is formulated with a dependency on the NULL prediction variable as follows:

  $$p(a_j|a_{j-1}, h_j; I) = \begin{cases} 1 & a_j = a_{j-1}, h_j = 0 \\ 0 & a_j \neq a_{j-1}, h_j = 0 \\ p_a(a_j|a_{j-1}; I) & h_j = 1 \end{cases} \tag{3.23}$$

  The phrase length model $n(\phi; e)$ is a form of English word fertility. It specifies the probability that an English word e generates a French phrase with ϕ words. A distribution $n(\phi; e)$ over the values $\phi = 1, \ldots, N$ is maintained as a table for each English word. The model also has a table of transition probabilities $p_a(i|i', I)$ for all English sentence lengths I. We use $d(0) = p_0$, and $d(1) = 1 - p_0$. p_0 is a tuning parameter that controls the tendency towards the insertion of French phrases.

- Word-to-Phrase translation: The translation of words to phrases is computed as follows:
$$Pr(v_1^K|a_1^K, h_1^K, \phi_1^K, K, J, \boldsymbol{e}) = \prod_{k=1}^{K} p(v_k|e_{a_k}, h_k, \phi_k) \tag{3.24}$$
  We introduce the notation $v_k = v_k[1], \ldots, v_k[\phi_k]$ and a dummy variable x_k:
$$x_k = \begin{cases} e_{a_k} & \text{if } h_k = 1 \\ NULL & \text{if } h_k = 0 \end{cases} \tag{3.25}$$
  where French phrases are conditionally independent given the individual English words. two models are introduced for word-to-phrase translation: The simplest model is based on context-independent word-to-word translation:
$$p(v_k|e_{a_k}, h_k, \phi_k) = \prod_{j=1}^{\phi_k} t_1(v_k[j]|x_k) \tag{3.26}$$
  A more complex model captures word context within the French language phrase via bigram translation probabilities:
$$p(v_k|e_{a_k}, h_k, \phi_k) = t_1(v_k[1]|x_k) \prod_{j=2}^{\phi_k} t_2(v_k[j]|v_k[j-1], x_k) \tag{3.27}$$

where $t_1(f|e)$ is the usual word-to-word translation probability and $t_2(f|f', e)$ is a bigram translation probability that specifies the likelihood that French word f' is followed by French word f in a phrase generated by English word e. In a nutshell, the parameter set θ of the WtoP HMM consists of the transition parameters p_a, the phrase length parameters n, the jumping-to-NULL parameter p_0, the unigram word-to-word translation parameters t_1 and the bigram translation probabilities t_2: $\theta = \{p_a(i|i', I), n(\phi, e), p_0, t_1(f|e), t_2(f|f', e)\}$.

The relationship between the current approach with the word-to-word HMM is straightforward. Constraining the phrase length model $n(\phi; e)$ to permit only phrases of one word gives a word-to-word HMM model. The extensions introduced in this method are the phrase count, phrase length model and the bigram translation model. The hallucination process addresses the NULL problem explained in section 3.2. The phrase length model is an alternative to fertility, and it is motivated by *stay* probabilities introduced in Toutanova et al. (2002). It is, however, more powerful than a single *stay* parameter.

WtoP HMM and IBM model 4 allow the same alignments. Both model NULL and fertility. However, distortion is not incorporated into the WtoP HMM model. Although WtoP model is more complex than the basic word-to-word HMM, the Baum-Welch and Viterbi algorithms can still be used. Hence, training can be done by forward-backward algorithm and by parallelizing we can control memory usage, reduce the time needed for training and increase the bitext for training.

3.6 HMM with Features

In this section, we discuss a method that shows how features can be added to a standard HMM model to inject prior knowledge to the model (Berg-Kirkpatrick et al., 2010). Each component multinomial of the generative model is turned into a miniature logistic regression model. The EM algorithm is used to learn the parameters. The E-step is unchanged, but the M-step involves gradient based training. Berg-Kirkpatrick et al. (2010) explained a general method to add features to an unsupervised model. We explain the feature-enhanced model for word alignment problem. To be consistent in the notations, we will use a slightly different notation from that of Berg-Kirkpatrick et al. (2010) for the word alignment task.

In section 3.6.2, we discuss discriminative feature-enhanced models developed recently. We briefly describe a discriminative variant of the Berg-Kirkpatrick et al. (2010) method and mention neural-network-based models for word alignment.

3.6.1 Feature-enhanced HMM

There are two types of distributions in the HMM model: emission and transition probabilities which are both multinomial probability distributions. Let the emission distribution $Pr(F_j = f_j|E_j = e_{a_j}, \theta)$ be parameterized by $\theta_{f,e}$ for each French word f given English word e. For word alignment, the transition probabilities are estimated based on the jump width as explained. However, the emission factors can be expressed as the output of a logistic regression model, replacing the explicit conditional probability table by a logistic function parameterized by weights $\mathbf{w}$ and features $\mathbf{h}$:

$$\theta_{f,e}(\mathbf{w}) = \frac{exp(\mathbf{w}.\mathbf{h}(f,e))}{\sum_{f'} exp(\mathbf{w}.\mathbf{h}(f',e))} \tag{3.28}$$

In the emission, the decision f is a French word and the context e is an English word. The denominator is a normalization term to make the factor a probability distribution over French word decisions. Here, $\mathbf{h}$ is a feature function consists of all indicator features on tuples (f, e), and $\mathbf{w}$ is the vector of weights associated to each feature that we want to optimize such that the likelihood of the data is maximized. As commonly used in log-linear models, the objective function is modified to include a regularization term:

$$L(\mathbf{w}) = \log Pr(F = \boldsymbol{f}|\mathbf{w}) - \kappa||\mathbf{w}||_2^2 \tag{3.29}$$

Note that the feature-based logistic expression, equation 3.28, is equivalent to the flat multinomial when the feature function $\mathbf{h}(f, e)$ consists of all the indicator features on tuples

(f, e), which we call BASIC features. The equivalence occurs when weights are set such that $w_{f,e} = log(\theta_{f,e})$. This is known as the natural parameterization of the multinomial distribution.

Optimization

EM algorithm is used to learn the parameters of the model. In the E-step, expected counts are calculated for each tuple of source word f and target word e:

$$\varepsilon_{f,e} \leftarrow \mathbb{E}_\theta\Big[\sum_{j \in J} \mathbb{1}(f_j = f, e_{a_j} = e | F = \boldsymbol{f})\Big] \tag{3.30}$$

In the M-step, to re-estimate the parameters, the expected counts are normalized as follows:

$$\theta_{f,e} \leftarrow \frac{\varepsilon_{f,e}}{\sum_{f'} \varepsilon_{f',e}} \tag{3.31}$$

Similarly, we can use EM to optimize $L(\mathbf{w})$ for the model with logistic parameterizations. The E-step precomputes θ parameters from $\mathbf{w}$ for each French word f and English word e using equation 3.28. Expected counts are computed using the forward-backward algorithm. The only difference from the standard model is that the conditional probabilities θ are now functions of $\mathbf{w}$. In the M-step, we use gradient based optimization. The goal is to find the $\mathbf{w}$ that maximizes the regularized log-likelihood:

$$\ell(\mathbf{w}, \boldsymbol{\varepsilon}) = \sum_{f,e} \varepsilon_{f,e} \log \theta_{f,e}(\mathbf{w}) - \kappa ||\mathbf{w}||_2^2 \tag{3.32}$$

Optimization of the objective function can be done using LBFGS. This method relies on the computation of the log-likelihood and the gradient at each step. The log-likelihood has the form given in equation 3.32, while the gradient with respect to $\mathbf{w}$ takes the following form:

$$\nabla \ell(\mathbf{w}, \boldsymbol{\varepsilon}) = \sum_{f,e} \varepsilon_{f,e} . \Delta_{f,e}(\mathbf{w}) - 2\kappa . \mathbf{w} \tag{3.33}$$

$$\Delta_{f,e}(\mathbf{w}) = \mathbf{h}(f, e) - \sum_{f'} \theta_{f',e}(\mathbf{w}) \mathbf{h}(f', e)$$

The M-step is now an iterative procedure which is much more expensive than before, because for each value of $\mathbf{w}$ considered during the search, $\theta(\mathbf{w})$ should be recomputed for each f and e. This computation is in proportion to the size of parameter space. Algorithm 3.6 shows the feature-enhanced EM algorithm.

Algorithm 3 Feature-enhanced EM

```
repeat
    compute expected counts $\boldsymbol{\varepsilon}$
    repeat
        compute $\ell(\mathbf{w}, \boldsymbol{\varepsilon})$
        compute $\nabla\ell(\mathbf{w}, \boldsymbol{\varepsilon})$
        $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \boldsymbol{\varepsilon}), \nabla\ell(\mathbf{w}, \boldsymbol{\varepsilon}))$
    until convergence
until convergence
```

Figure 3.6: Feature-enhanced EM

Word Alignment Features

The BASIC features provide a strong baseline performance. Table 3.1 shows linguistically-motivated features that are added to the model in Berg-Kirkpatrick et al. (2010) in order to inject prior knowledge. These features are designed for Chinese-English. However, language-specific features can be designed for other language pairs similarly. Note that all the features in this table are binary.

Table 3.1: Feature templates for experiments

Template	Description
`BASIC`	fires for each source word and target word pair in the alignment
`EDIT-DISTANCE`	fires when the edit distance between source word and the target word is a specific value
`DICTIONARY`	fires when the source word and the target word pair is in the dictionary
`STEM`	fires for each source word and stem of the target word pair (for porter stemmer)
`PREFIX`	fires for each pair of source word and prefix of length 4 of the target word
`CHARACTER`	fires for each target word and ith source (Chinese here) character pair

3.6.2 Discriminative methods

A discriminative log-linear variant of the Berg-Kirkpatrick et al. (2010) method is introduced by Dyer et al. (2011). This method can incorporate arbitrary overlapping features, and it is used to infer word alignment. A log-linear model with parameter $\mathbf{w}$ and feature function $\mathbf{h}$ is used to model $p(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}, J)$ directly. The feature function used in this model includes word association features, positional features, lexical features and HMM-like path features. As for the inference, Dyer et al. (2011), design their features to keep the width of tree decomposition low to allow exact inference with dynamic programming. To learn the parameters, we select $\mathbf{w}$ that minimizes the ℓ_1 regularized conditional log-likelihood. Dyer et al. (2011) use an online method that approximates ℓ_1 regularization and only depends on the gradient of the unregularized objective (Tsuruoka et al., 2009).

Blunsom and Cohn (2006) use a Conditional Random Field (CRF), a discriminative model, on a small supervised setting. The model directly encodes $p(\boldsymbol{a}|\boldsymbol{f}, \boldsymbol{e})$ with a CRF. With a first order Markov assumption, exact inference and efficient learning are possible using forward-backward and Viterbi algorithms.

Yang et al. (2013) propose a model that integrates a multi-layer neural network into an HMM-like framework. They adapt and extend the context dependent deep neural network HMM (CD-DNN-HMM) (Dahl et al., 2012) model to the HMM-based word alignment model. In their method, context dependent lexical translation score is computed by neural network, and distortion is modelled by a simple jump distance scheme. The model is discriminatively trained on bilingual corpus, in a supervised setting, while huge monolingual data is used to train word embeddings.

Tamura et al. (2014) propose a word alignment model based on a recurrent neural network (RNN) to extend the feed-forward neural network (FNN) model of Yang et al. (2013). This RNN-based model captures long alignment history through recurrent architectures. Compared to the CD-DNN-HMM approach which can only explore the context in a window, the RNN predicts the j-th alignment a_j by conditioning on all the preceding alignments a_1^{j-1}. Moreover, noise-contrastive estimation (NCE) is applied for unsupervised training of the model.

3.7 Summary

In this chapter, we have categorized the extensions to the basic HMM-based model into six groups of models based on the enhancement they offer: refined alignment (transition) models (section 3.1), NULL-enhanced models (section 3.2), fertility-enhanced models (section 3.3), part-of-speech-enhanced model (section 3.4), word-to-phrase model (3.5) and feature-enhanced model (section 3.6). We examine whether these extensions can improve the alignment quality in the next chapter.

Chapter 4

Results and Analysis

In this chapter, we look at the results obtained by the methods discussed in the previous chapter and we analyze the effect of each extension to the baseline model. The methods we have seen measure the quality of their alignment using alignment error rate (AER). Therefore, we explain this measure in the next section.

4.1 Evaluation with AER

The quality of the alignment methods is evaluated by the performance on a test set for which a gold standard has been established by human annotators. The annotators are asked to specify alignments of two kinds: an S (sure) alignment, for alignments that are unambiguous and a P (possible) alignment, for ambiguous alignments. Thus, the reference alignment obtained may contain many-to-one and one-to-many relationships. The quality of an alignment $A = \{(j, a_j) | a_j > 0\}$ is evaluated using a measure called alignment error rate (AER), which is defined as

$$AER = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} \tag{4.1}$$

4.2 HMM with Empty Word

Och and Ney (2000a) perform their experiments on a German-English training corpus of 34K sentences and a test corpus of 354 sentences. They provide a comparison of IBM models 1-4 with the HMM alignment model. They find that more sophisticated models perform better. The best performance is achieved by Model 4. On the other hand, the HMM model outperforms IBM Models 1-3, which confirms the importance of first-order alignment models. They further analyze the effect of modelling an empty word and report improvements in alignment quality.

Toutanova et al. (2002) introduce an alternative model for addressing the empty word problem. They perform experiments on French-English Hansards corpus and a manually aligned test data of 400 sentences. They experiment with training data of sizes ranging from 5K to 50K. The new NULL model, called **Null**, outperforms the baseline HMM at every data size; larger error reduction is reported for bigger training sets (up to 9.2% error reduction).

4.3 Refined Alignment Models

Och and Ney (2000a) conduct an experiment to show the effect of dependence on word classes in the HMM and IBM Model 4. Using word classes improves the results of HMM by 0.9% and Model 4 by 0.5%.

He (2007) evaluate the word-dependent transition model for HMM on the French-English Hansards corpus. A subset of 500k sentences are used including 447 test sentence-pairs. Compared to the basic HMM, word-dependent HMM reduce AER by more than 13%. This model even outperforms IBM Model 4 when two directions $French \rightarrow English$ and $English \rightarrow French$ are combined. The method is further evaluated on machine translation test and is shown to significantly improve the baseline HMM in terms of BLEU (Papineni et al., 2002).

4.4 Fertility and POS Tags for HMM

Apart from the **Null** model mentioned earlier, Toutanova et al. (2002) provide two other extensions over the baseline HMM, which are called **Tags** and **SG**, referring to the POS-enhanced HMM model and fertility-enhanced HMM model, respectively. The **SG** model reduces AER by up to 10% which is a significant improvement. The model **Tags** reduces AER for the smallest dataset by 7.6%. Toutanova et al. (2002) note that the combination of **Tags** and **SG** or **Null** outperforms the individual model in the combination since they address different problems that arise orthogonally. For instance, the combination of **SG** and **Tags** reduces AER by 16% and **SG + Null** reduces AER by up to 12.3%. All of these error reductions are statistically significant at the $\sigma = 0.05$ confidence level according to the paired t-test.

Figure 4.1 shows learning curve for three models: **Och**, **Tags** and **Null+Tags**. **Och** is the baseline HMM model of Och and Ney (2000a). To obtain the results for the **Och** model, GIZA++ was run. The **Tags** model outperforms the **Och** model for smaller training data sets. As the training size increases, the **Och** model catches up with the **Tags** model and even outperforms it. **Tags** model, therefore, can be used when large amounts of parallel data are not available. The third curve in the figure, **Null+Tags** model, significantly outperforms the **Och** model.

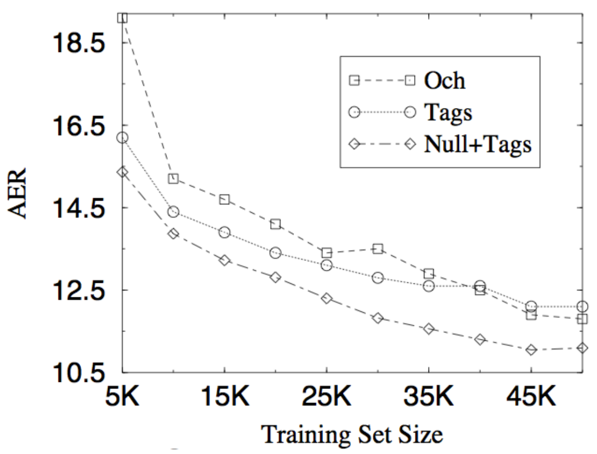


Figure 4.1: Learning curves for three models: **Och**, **Tags** and **Null+Tags**

4.5 Feature-enhanced Model

Berg-Kirkpatrick et al. (2010) experiment on the standard hand-aligned portion of NIST 2002 Chinese-English development set and measure alignment quality using AER. They train the model on 10K sentences of FBIS Chinese-English newswire. The alignments in the baseline and the feature-enhanced HMM are computed by training the models in both directions, combining the results with hard union competitive thresholding (DeNero and Klein, 2007), and agreement training for the HMM (Liang et al., 2006). Berg-Kirkpatrick et al. (2010) find that the feature-enhanced HMM outperforms baseline HMM by 3.8 AER.

4.6 Summary and Discussion

In this chapter, we have examined the results of the models discussed in the preceding chapter. We looked at the most prevalent measure used for alignment quality. We also considered the extent each method could improve the baseline HMM. Modelling the empty word is definitely helpful for HMM. Moreover, a more refined alignment model and modelling fertility improve the alignment quality. Adding part-of-speech tag information to the HMM can be helpful when large amounts of parallel data are not available. All the methods discussed use only European languages except the feature-enhanced HMM (Berg-Kirkpatrick et al., 2010) which does experiment over Chinese-English data. It could be the case that their method does not improve the baseline HMM for the European language pairs. Although feature-enhanced model is a very powerful and interesting method, it suffers from scalability issues for large datasets.

Chapter 5

Conclusion

There have been many years of research on word alignment. HMM-based word alignment model has remained to be one of the most widely studied models because it can be extended to perform very close to IBM Model 4, it is easy to implement and modify and time-efficient to train.

In this report, we examined the HMM-based word alignment model for SMT. We discussed the motivation for using HMM for word alignment. After presenting the basic HMM-based model, we looked at methods that extend this basic model. We first explored methods that used refined alignment models (transition probabilities) in the HMM. We also discussed methods that model empty word and fertility for HMM. We showed how part-of-speech tag information can be incorporated into the basic HMM. A word-to-phrase HMM, an extension of word-to-word HMM, is also discussed. The last method we discussed incorporates features into a standard HMM model to inject prior knowledge to the model. We examined the results of the methods and analysed the effect of each extension to the baseline model.

Bibliography

Leonard E Baum. An equality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3:1–8, 1972.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590. Association for Computational Linguistics, 2010.

Phil Blunsom and Trevor Cohn. Discriminative word alignment with conditional random fields. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 65–72. Association for Computational Linguistics, 2006.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

John DeNero and Dan Klein. Tailoring word alignments to syntactic machine translation. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 17, 2007.

Yonggang Deng and William Byrne. Hmm word and phrase alignment for statistical machine translation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(3):494–507, 2008.

Chris Dyer, Jonathan Clark, Alon Lavie, and Noah A Smith. Unsupervised word alignment with arbitrary features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 409–419. Association for Computational Linguistics, 2011.

Jean-Luc Gauvain and Chin-Hui Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *Speech and audio processing, ieee transactions on*, 2(2):291–298, 1994.

Xiaodong He. Using word dependent transition models in hmm based word alignment for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 80–87. Association for Computational Linguistics, 2007.

Reinhard Kneser and Hermann Ney. Forming word classes by statistical clustering for statistical language modelling. In *Contributions to Quantitative Linguistics*, pages 221–226. Springer, 1993.

Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics, 2006.

Franz Josef Och and Hermann Ney. A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 1086–1090. Association for Computational Linguistics, 2000a.

Franz Josef Och and Hermann Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 440–447. Association for Computational Linguistics, 2000b.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51, 2003.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. Recurrent neural networks for word alignment model. In *Proceedings of EMNLP*, 2014.

Kristina Toutanova, H Tolga Ilhan, and Christopher D Manning. Extensions to hmm-based statistical word alignment models. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 87–94. Association for Computational Linguistics, 2002.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 477–485. Association for Computational Linguistics, 2009.

Andrew J Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, 1967.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics, 1996.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. Word alignment modeling with context dependent deep neural network. In *ACL (1)*, pages 166–175, 2013.

Shaojun Zhao and Daniel Gildea. A fast fertility hidden markov model for word alignment using mcmc. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 596–605. Association for Computational Linguistics, 2010.