

# An Efficient Phrase-to-Phrase Alignment Model for Arbitrarily Long Phrase and Large Corpora

Ying Zhang Stephan Vogel

Language Technologies Institute  
School of Computer Science  
Carnegie Mellon University  
{joy+,vogel+}@cs.cmu.edu

**Abstract.** Most statistical machine translation (SMT) systems use phrase-to-phrase translations to capture local context information, leading to better lexical choices and more reliable word reordering. Long phrases capture more contexts than short phrases and result in better translation qualities. On the other hand, the increasing amount of bilingual data poses serious problems for storing all possible phrases. In this paper, we describe a novel phrase-to-phrase alignment model which allows for arbitrarily long phrases and works for very large bilingual corpora. This model is very efficient in both time and space and the resulting translations are better than the state-of-the-art systems.

## 1. Introduction

In recent years, various phrase-to-phrase translation models (Och 1999; Marcu & Wong 2002; Koehn 2003; Zhang 2003) have shown great advantages over the word-based systems (Brown 1990). We believe that longer phrases encapsulate more contexts of the words and the translation qualities are expected to be higher than that of short phrases. Unfortunately, given the increasing volume of the parallel bilingual data for some major languages such as Arabic and Chinese, storing and loading all possible phrase translations from the training corpus becomes more and more expensive by means of space and time in computation. To keep the phrasal translation model of a reasonable size, some models (Koehn 2003) and (Zhang 2003) limit the length of the phrases to be no more than 3 words while others (Vogel 2003) sub-samples the training corpus based on the testing data to down-scale the problem. In this paper, we introduce a new strategy to cope with this problem. Instead of aligning the phrases offline, we extract the phrase translations on the fly for each testing sentences. We use suffix array (Manber 1990) to index the training corpus and a novel fast algorithm to search all the

substrings (phrases) of the testing sentences in the training data. For each sentence pairs that contain the phrases in the testing sentence, a new phrase alignment model, Alignment via Sentence Partition (*ASP*) is used to extract the translations for the phrase. Thus, we do not need to store any phrase translations and we can use arbitrarily long phrases.

In the following sections, we first show the empirical evidence that long phrases do improve the translation qualities. Then we will introduce our phrase alignment model *ASP* which finds the alignment for a source phrase of any length. The suffix array and the fast search algorithm, the key components that enables this approach to be feasible are discussed in details in section 4. In the end, we will introduce a mixture online/offline alignment strategy which allows for arbitrarily long phrases and works with arbitrarily large bilingual corpora efficiently.

## 2. Phrase Length vs. Translation Quality

Throughout this paper, TIDES Chinese-English bilingual corpora are used as the training data and all experiments are tested on three years' TIDES/NIST MT evaluation set. Yet, the

approach described in this paper is language independent and can be applied to other language pairs. Table 1 lists the statistics of the training and the testing corpora, including the number of words ( $N$ ), total number of sentences ( $Sent.$ ) and the averaged length of each sentence ( $Avg. m$ )

Corpus		N	Sent.	Avg. $m$
Training	FBIS.gb	4.6M	128K	36.3
	UN.gb	60.0M	1.9M	31.3
Testing	TIDES02	24.3K	878	27.7
	TIDES03	26.2K	919	28.5
	TIDES04	52.2K	1788	29.2

Table 1. Corpus statistics

First, we analyzed the  $n$ -gram coverage of the testing data given the training corpus. Table 2 shows the  $n$ -gram coverage of the TIDES04 data given two training corpora. On the word level (unigram), both training corpora cover the testing data well. More than 99% of words in the testing data can be found in either training set. On the other hand, the coverage for long phrases decreases rapidly, less than 5% of 5-grams in the testing data occur in the training data. Still, it is worth noticing that there is a significant number of long phrases that are covered in the training data and even one 68-gram occurred in the FBIS training data.

n	TIDES04			
	FBIS		UN	
1	51767	99.2%	51848	99.3%
2	36758	72.9%	40008	79.4%
3	16066	33.0%	19236	39.6%
4	5702	12.2%	6438	13.7%
5	2155	4.8%	1845	4.1%
15	172	0.6%	1	0.0%
68	1	0.3%	-	-

Table 2. Training data  $n$ -gram coverage of the TIDES04 testing data

We did a series of controlled experiments (Table 3) to study how translation qualities are affected by the length of phrases in the translation model. FBIS data was used as the training set and the translation model is trained by the Alignment via Sentence Partition (*ASP*) algorithm described in the next section.

Max. Phrase Len.	Modified $n$ -gram Precision (%)					NIST5	BLEU4
	1	2	3	5	8		
1	59.67	19.66	5.93	0.70	0.02	6.5689	0.1057
2	64.57	25.06	9.33	1.17	0.03	7.1641	0.1450
3	65.82	26.91	10.95	1.62	0.04	7.3846	0.1642
5	66.04	27.48	11.47	1.81	0.11	7.4347	0.1701
10	66.04	27.47	11.45	1.84	0.10	7.4374	0.1755

Table 3. Phrase Length vs. Translation Quality. FBIS data for training and tested on TIDES02

We restrained the longest phrase allowed to be one word (word-to-word translation model), two words and so on. Each translation model was then used by a decoder (Vogel et al. 2003) which searches for the best hypothesis that maximizes the translation score. The translations are compared against 4 human reference translations using the BLEU (Papineni 2002) and the NIST MTEval metrics (NIST).

From this result, we observe that going from the word-to-word translation model to a simple two-word phrasal model improved the translation significantly (+9% on NIST and +37% on BLEU). Longer phrases result in higher BLEU/NIST scores, i.e., better translation qualities. When using phrases longer than 5 words, the BLEU score improved from 0.1701 to 0.1755 for about 3%. The effects of allowing long phrases in the translation model should have been more prominent if the evaluation metrics are more sensitive to long  $n$ -gram matchings. It has been noted in (Zhang 2004) that 80% of the NIST score comes from the matches of the unigrams, most of the matched 5-grams are given no credit in the final NIST score. BLEU scores reported in Table 3 were capped to  $n$ -gram precisions at 4-grams, thus only gave credits for long  $n$ -gram matches indirectly.

From the above analysis, we conclude that long phrases in the translation model improve the translation quality. In the following sections, we will describe an efficient phrase alignment model that allows arbitrarily long phrases in the TM. First, we will introduce our phrase-to-phrase alignment model.

### 3. Phrase-to-phrase Alignment via Sentence Partition

Let  $C$  be a bilingual corpus consists of  $S$  sentence pairs. Denote  $C_f = \{f_1, f_2, \dots, f_s, \dots, f_S\}$  for the source side of  $C$  and  $C_e = \{e_1, e_2, \dots, e_s, \dots, e_S\}$  for the target side. In  $C$ , sentence  $e_s$  and  $f_s$  are translations of each other.

Assuming that we are searching for a good translation for one source phrase  $f = f_1 f_2 \dots f_m$ , and we find a sentence in the bilingual corpus, which contains this phrase. We are now interested in finding a sequence of words  $e = e_1 e_2 \dots e_l$  in the target sentence, which is an optimal translation of the source phrase. Any sequence of words in the target sentence is a translation candidate, but most of them will not be considered as translations of the source phrase at all, whereas some can be considered as partially correct translations, and a small number of candidates will be considered as acceptable or good translations. We want to find these good candidates.

#### 3.1. Constrained Word Alignment

The IBM1 word alignment model aligns each source word to all target words with varying probabilities. Typically, only one or two words will have a high alignment probability, which for the IBM1 model is just the lexicon probability. We now modify the IBM1 alignment model by not summing the lexicon probabilities of all target words, but by restricting this summation in the following ways:

- For words inside the source phrase we sum only over the probabilities for words inside the target phrase candidate, and for words outside of the source phrase we sum only over the probabilities for the words outside the target phrase candidates;
- The position alignment probability, which for the standard IBM1 alignment is  $1/l$ , where  $l$  is the number of words in the target sentence, is modified to  $1/l$  inside the source phrase and to  $1/(l-1)$  outside the source phrase.

More formally, we calculate the constrained alignment probability as:

$$\begin{aligned} & p_{i_1, i_2}(f | e) \\ &= \prod_{j=1}^{i_1-1} \sum_{i \in (i_1, \dots, i_2)} \frac{1}{l-1} p(f_j | e_i) \\ &\times \prod_{j=i_1}^{j_2} \sum_{i=i_1}^{i_2} \frac{1}{k} p(f_j | e_i) \\ &\times \prod_{j=j_2+1}^J \sum_{i \in (i_1, \dots, i_2)} \frac{1}{l-1} p(f_j | e_i) \end{aligned}$$

and optimize over the target side boundaries  $i_1$  and  $i_2$ .

$$(i_1, i_2) = \arg \max_{i_1, i_2} \{p_{i_1, i_2}(f | e)\}$$

It should be mentioned that the left segment or the right segment or both segments can be empty. The alignment calculation is then accordingly modified. This means also that the entire sentence can be used as a phrase, which is then alignment to the entire target sentence.

#### 3.2. Looking from both Sides

It is well known that „looking from both sides” is better than calculating the alignment only in one direction, as the word alignment models are asymmetric with respect to aligning one to many words. Similar to  $p_{i_1, i_2}(f | e)$  we can calculate  $p_{i_1, i_2}(e | f)$ , now summing over the source words and multiplying along the target words.

To find the optimal target phrase we interpolate both alignment probabilities and take the pair  $(i_1, i_2)$  which gives the highest probability

$$(i_1, i_2) = \arg \max_{i_1, i_2} \{(1-c)p_{(i_1, i_2)}(f | e) + c \cdot p_{(i_1, i_2)}(e | f)\}$$

It should also be mentioned that single source words are treated in the same way, i.e. just as phrases of length  $l$ . The target translation can then be one or several words.

### 4. Locating Source Phrases in the Bilingual Corpus using Suffix Array

Enumerating all the phrases in the training corpus and find their alignment via *ASP* is almost impossible considering the number of phrases of any length in a corpus. Table 4 gives the statistics of phrase numbers in the FBIS Chinese-English corpus.

n	Num of Types (uniq. n-grams)	Num of Tokens
1-gram	33,554	4,646,656
2-gram	806,201	4,518,690
3-gram	2,277,682	4,390,724
4-gram	3,119,107	4,262,867
5-gram	3,447,066	4,135,067
6-gram	3,546,095	4,008,201

Table 4. *n*-gram statistics for FBIS training data

In the offline TM training approach, where one enumerates all the source phrases in the bilingual corpus and extracts their possible translations, it is clear that one better not to store translations for phrases longer than 3 words, otherwise the decoder is not able to load the phrase translation model during decoding. To benefit from the longer phrase matching, we introduce the online phrase extracting approach using the suffix array.

#### 4.1. Suffix Array

Suffix array was introduced as an efficient method to find instances for a string in a large text corpus. It has been successfully applied in many natural language processing areas (Yamamoto 2001) and (Ando and Lee 2003).

For a monolingual text  $C_f$  with  $N$  words, represent it as a stream of words:  $a_0 a_1 \dots a_N$ . Denote by  $A_i = a_i a_{i+1} \dots a_N$  the suffix of  $C_f$  that starts at position  $i$ . The suffix array of  $C_f$  is a sorted array,  $Pos$ , of all suffixes of  $C_f$ , namely,  $Pos[k]$  is the starting position of the  $k$ -th smallest suffix in the set  $\{A_0, A_1, \dots, A_N\}$ , or in other words,  $A_{pos[0]} < A_{pos[1]} < \dots < A_{pos[N]}$ , where " $<$ " denotes the lexicographical order. Figure 1 gives a simple example of the suffix array.

Text $C_f$	$a_0$ =finance	$a_1$ =is	$a_2$ =the	$a_3$ =core	$a_4$ =of	$a_5$ =the	$a_6$ =economy
Suffixes:	$A_0$ "finance is the core of the economy" $A_1$ "is the core of the economy" $A_2$ "the core of the economy" $A_3$ "core of the economy" $A_4$ "of the economy" $A_5$ "the economy" $A_6$ "economy"						
Because:	$A_3$ "core..." < $A_6$ "economy" < $A_0$ "finance..." $A_1$ "is..." < $A_4$ "of..." < $A_2$ "the core..." < $A_5$ "the economy..."						
Sorted Suffixes $Pos$ :	Index    0   1   2   3   4   5   6 Pos[index] 3   6   0   1   4   2   5						

Figure 1. Indexing the corpus using the Suffix Array

The sorting of set  $\{A_0, A_1, \dots, A_N\}$  can be done in  $\log_2(N+1)$  stages and requires

$O(N \log N)$  time in the worst case (Manber 1993). Table 5 shows the time needed to sort the suffix array for the training corpora.

Corpus	Words	Time
FBIS.gb	4.6M	95.4s
UN.gb	60.0M	5423.7s

Table 5. Time needed to sort the suffix array

Given a string  $f = f_1, f_2, \dots, f_i, \dots, f_m$ , we want to locate all the substrings of  $f$  in corpus  $C_f$ . There are  $m$  unigram,  $m-1$  bigram,  $m-2$  trigram, ... , and 1  $m$ -gram in  $f$ . Based on the original algorithm described in (Manber 1990) locating one  $n$ -gram in  $C_f$  requires  $O(n \log N)$  because of  $O(n)$  single word comparison for each of the  $O(\log N)$  binary searches. A naive algorithm (Figure 2) thus requires:

$$\sum_{m=1}^m (m-n+1) \cdot n \log N = \frac{m^3 + 3m^2 + 2m}{6} \log N,$$

which is  $O(m^3 \log N)$  in time. We show next that the run time can be greatly reduced in the binary search and the  $n$ -gram comparisons using our novel fast search algorithm.

```

Input: string  $f = f_1, f_2, \dots, f_i, \dots, f_m$ 
1 for  $n \leftarrow 1$  to  $m$  do
2   for  $i \leftarrow 1$  to  $(m-n+1)$  do
3     search substring  $f_i^{i+n-1}$  in  $C_f$  with
        $O(n \log N)$  in time;
4   end
5 end

```

Figure 2. A naive Algorithm for Searching all Substrings

#### 4.2. Fast Algorithm for Searching Substrings

The fast substring searching algorithm is based on the following theorems<sup>1</sup>:

**Definition** For a string  $u$ , let  $u^p$  be the prefix consisting of the first  $p$  words of  $u$  if  $u$  contains more than  $p$  words and  $u$  otherwise. Define the relation  $<_p$  to be the lexicographical order of  $p$ -word prefixes; that is,  $u <_p v$  iff  $u_1^p < v_1^p$ . Relations  $\leq_p, =_p, >_p$ , and  $\geq_p$  are defined in a similar way.

**Definition** Define the lexicographical relation  $\mathcal{R}$  as a function. For two words  $u$  and  $v$ ,  $\mathcal{R}(u, v) \in \{<, \leq, =, \geq, >, <_p, \leq_p, =_p, \geq_p, >_p\}$ .

<sup>1</sup> The proof of the theorems are trivial and not given in the paper.

**Theorem 1** Substring  $f_i^{i+n-1}$  has occurrences in  $C_f$  only if both substring  $f_i^{i+n-2}$  and  $f_{i+1}^{i+n-1}$  exist in  $C_f$ .

For example, in string  $f =$  "consumer shopping is the core of economy", substring "is the core of economy" could occur in  $C_f$  only if both substrings  $f' =$  "is the core of" and  $f'' =$  "the core of economy" occur in  $C_f$ . Or in other words, if we know that either  $f'$  or  $f''$  has no occurrences in  $C_f$ , we do not need to search for the occurrences of  $f$ .

**Theorem 2** If substring  $f_i^{i+n-1}$  and  $f_i^{i+n}$  both exist in  $C_f$ , let,

$$l = \min(k : f_i^{i+n-1} \leq_n A_{Pos[k]} \text{ or } k = N),$$

$$r = \max(k : A_{Pos[k]} \leq_n f_i^{i+n-1} \text{ or } k = -1),$$

and

$$l' = \min(k : f_i^{i+n} \leq_{n+1} A_{Pos[k]} \text{ or } k = N),$$

$$r' = \max(k : A_{Pos[k]} \leq_{n+1} f_i^{i+n} \text{ or } k = -1),$$

then  $[l', r'] \subseteq [l, r]$ .

From Theorem 1 we know that substring "the economy" could occur in  $C_f$  only when "the" occurs. Theorem 2 further states that if we know the index range for "the" in  $Pos$  array is [5, 6], the index range for "the economy" has to be a subset of [5, 6]. In other words, the index range of "the" in the suffix array narrows down the search range for phrase "the economy".

**Theorem 3** If string  $u_1^{r_1}$  and  $v_1^{r_2}$  have common prefix of length(LCP)  $p$ , i.e.  $u_1^{p-1} = v_1^{p-1}$ , then  $\mathcal{R}(u_1^p, v_1^p) = \mathcal{R}(u_p, v_p)$ .

Suppose that we want to search locations for phrase "the economy" in  $C_f$ . After one binary search, we have found that all the suffixes start with "the" are in the range of [5, 6]. This means that all the suffixes in this range have the same prefix "the" (LCP=1). Following Theorem 2, we will search inside the range [5, 6] for the occurrences of "the economy". Theorem 3 states that in doing so, one does not need to compare each suffix in the range with the phrase "the economy" since we know they have the same prefix "the", instead, only the next word needs to be compared with "economy" to determine the lexicographical relation between the query phrase and the suffix.

Based on these three theorems, we developed a very fast algorithm for searching all substrings of a phrase (Figure 3 and Figure 4). In this algorithm, three two-dimensional matrices  $L_{m \times m}$ ,  $R_{m \times m}$  and  $Q_{m \times m}$  are used.  $L[i, n]$ ,  $R[i, n]$  and  $Q[i, n]$  contain matching

information for substring  $f_i^{i+n-1}$ , i.e. the substring starting from  $f_i$  with length  $n$ . Let,

$$L[i, n] = \min(k : f^{i+n-1} \leq_n A_{Pos[k]} \text{ or } k = N)$$

$$R[i, n] = \max(k : A_{Pos[k]} \leq_n f^{i+n-1} \text{ or } k = -1)$$

and  $Q[i, n] = 1$  if there are at least one occurrences of  $f_i^{i+n-1}$  in  $C_f$  and 0 for none. Informally,  $L[i, n]$ ,  $R[i, n]$  stores the index range of substring  $f_i^{i+n-1}$  in the  $Pos$  array, and  $Q[i, n]$  is a Boolean indicator of the existence of the substring.

**Input:** string  $f = f_1, f_2, \dots, f_i, \dots, f_m$

- 1 **Initialize** Initialize  $L[1, *]$ ,  $R[1, *]$  and  $Q[1, *]$ ;
- 2 **for**  $n \leftarrow 2$  **to**  $m$  **do**
- 3     **for**  $i \leftarrow 1$  **to**  $(m - n + 1)$  **do**
- 5         **if** ( $Q[n - 1, i] = 1$  &  
           $Q[n - 1, i + 1] = 1$ ) **then**
- 7              $(k'_l, k'_r) \leftarrow$   
              SearchStringInRange( $f_{i+n-1}$ ,  
               $n - 1, L[n - 1, i], R[n - 1, i]$ );
- 8             **if**  $p_l \leq p_r$ , **then**
- 9                  $L[n, i] \leftarrow k'_l$ ;
- 10                 $R[n, i] \leftarrow k'_r$ ;
- 11                 $Q[n, i] \leftarrow 1$ ;
- 12             **else**
- 13                  $Q[n, i] \leftarrow 0$ ;
- 14             **end**
- 15             **else**
- 16                  $Q[n, i] \leftarrow 0$ ;
- 17             **end**
- 18         **end**
- 19 **end**

Figure 3: Fast search for all substrings (main routine)

**Input:** word  $w$ ; int  $lcp$ ; index of  $Pos$   $k_l, k_r$

**Result:**  $k'_l, k'_r$

- 1  $k'_l \leftarrow k_l$ ;
- 2  $k'_r \leftarrow k_r$ ;
- 3 **while**  $k'_r - k'_l > 1$  **do**
- 4      $k''_m \leftarrow (k'_l + k'_r) / 2$ ;
- 5     **if**  $w \leq A_{Pos[k''_m] + lcp}$  **then**
- 7          $k'_r \leftarrow k''_m$ ;
- 8     **else**
- 9          $k'_l \leftarrow k''_m$ ;
- 10     **end**
- 11 **end**
- 12  $k'_l \leftarrow k'_l$ ;
- 13  $k'_l \leftarrow k_l$ ;
- 14  $k'_r \leftarrow k_r$ ;
- 15 **while**  $k'_r - k'_l > 1$  **do**
- 16      $k''_m \leftarrow (k'_l + k'_r) / 2$ ;
- 18     **if**  $w \geq A_{Pos[k''_m] + lcp}$  **then**
- 19          $k'_l \leftarrow k''_m$ ;
- 20     **else**
- 21          $k'_r \leftarrow k''_m$ ;
- 22     **end**
- 23 **end**
- 24  $k'_r \leftarrow k'_l$ ;
- 25 **return**  $k'_l, k'_r$

Figure 4: Subroutine of SearchStringInRange with LCP using the binary search

Figure 5 shows a simple example of locating all the substrings of "growth is the essence of the economy" in  $C_f$  "finance is the core of the economy".

"growth" [N,-1] Q=0	"is" [3,3] Q=1	"the" [4,5] Q=1	"essence" [N,-1] Q=0	"of" [4,4] Q=1	the [5,6] Q=1	economy [1,1] Q=1
"growth is" Q=0	"is the" [3,3] Q=1	"the essence" Q=0	"essence of" Q=0	"of the" [4,4] Q=1	"the economy" [6,6] Q=1	
"growth is the" Q=0	"is the essence" Q=0	"the essence of" Q=0	"essence of the" Q=0	"of the economy" [4,4] Q=1		
...	...	...	...			

**Figure 5. A simple example of the fast algorithm.**  
Shown for each cell are the corresponding substring,  
the [L, R] and Q values

Another way to look at this is that the fast algorithm actually executes  $m$  naive searches for the exact substring  $f_i^m$  ( $i=1, \dots, m$ ) and along the trace of the binary search, bookkeeping the occurrence ranges for its prefixes. Each search uses  $O(\log N)$  comparisons and each such comparison requires only one word comparison. Thus the search  $f_i^m$  is of complexity  $O(\log N)$ . The time complexity is then  $O(m \cdot \log N)$  for searching all the substrings in sentence  $f$ . Table 6 compares the native algorithm and the fast algorithm over the time needed to search all the substrings of the testing sentences in the training corpora. It is obvious to see the speed up of the fast algorithm.

Testing	Training	Naive Alg.	Fast Alg.
TIDES2002	FBIS.gb	18	<b>0.27</b>
	UN.gb	198	<b>0.39</b>
TIDES2003	FBIS.gb	19	<b>0.28</b>
	UN.gb	205	<b>0.41</b>
TIDES2004	FBIS.gb	42	<b>0.58</b>
	UN.gb	465	<b>0.84</b>

**Table 6. Time needed to search all substrings of the testing sentences in the training data. All the experiments are on a machine with CPU 3.20GHz and 3.7G RAM running Linux.**

### 4.3. Retrieving Sentence ID and Position Offset of a Phrase in the Corpus

The primary motivation of the fast substring searching algorithm was to efficiently locate all substrings of a testing sentence in the training corpus, so that the alignment program *ASP* can

extract the corresponding translations from the target side of the bilingual corpus based on the sentence number and the position in the sentence.

If the index  $pos$  of the  $Pos$  array is in the range of  $(L[i, n], R[i, n])$ , then the string  $a_{pos}a_{pos+1} \dots a_{pos+n-1}$  in  $C_f$  equals to  $f_i^{i+n-1}$ . Now, we need to convert the absolute position  $pos$  in the sorted suffix array to a tuple  $\langle s, d \rangle$  such that the substring from  $d$  to  $d+n-1$  in  $f_s = f_i^{i+n-1}$ . Then we can extract the alignment from the sentence pair  $(f_s, e_s)$  using the *ASP* algorithm.

Define vocabulary  $\mathcal{F}$  as a mapping from strings to integers. For sentence  $f_s = f_1, f_2, \dots, f_i, \dots, f_m$ ,  $\mathcal{F}(f_s)$  is the short form for  $\mathcal{F}(f_1), \mathcal{F}(f_2), \dots, \mathcal{F}(f_m)$ , i.e. mapping all the words in sentence  $f_s$  to their corresponding vocabulary IDs. Insert the sentence boundary marker  $[eos]$  and sentence ID to the beginning of each sentence. Using the vocabulary  $\mathcal{F}$ , we can convert  $C_f$  to a stream of integers  $C_f' = 1, \mathcal{F}(f_1), \mathcal{F}([eos]), 2, \mathcal{F}(f_2), \mathcal{F}([eos]), \dots, S, \mathcal{F}(f_s)$ .  $C_f'$  is then indexed and searched in the same way as in the previous sections. When a position in the suffix array  $pos$  is found a the location of a phrase, apply the algorithm in Figure 6 to convert it to  $\langle s, d \rangle$ .

**Input:**  $pos$

- 1  $d = 0;$
- 2 **while**  $a_{pos} \neq \mathcal{F}([eos])$  **do**
- 3      $pos \leftarrow pos - 1;$
- 4      $d \leftarrow d + 1;$
- 5 **end**
- 6  $s \leftarrow a_{pos-1};$
- 7 **return**  $s, d$

**Figure 6. Retrieving sentence ID and position offset**

## 5. Mixture Online/Offline Alignment Model

Given a testing sentence  $f$ , three steps need to be done before the translation lattice can be built. First, we need to construct the search matrices  $L$ ,  $R$  and  $Q$  to locate the occurrences of all the substrings in  $f$ . This costs  $O(m \cdot \log N)$  for each testing sentence. Then we need to locate the sentence ID and the position offset for each occurrence. The averaged time is  $\bar{m}/2$  for each occurrences, where  $\bar{m}$  is the averaged sentence length in  $C_f$ . In the end, we apply the *ASP* algorithm for each sentence pair found. Table 7 shows the number of  $n$ -grams in the TIDES03

testing data which can be matched in the FBIS.gb training data and the total occurrences of the matched  $n$ -gram in the training data.

n	TIDES03 on FBIS.gb	
	matched $n$ -grams in test	total occurrences of matched $n$ -gram in training
1	25804	1046129684
2	16283	5894205
3	5654	297155
4	1482	16764
5	403	1750
14	9	30
19	1	2

Table 7. Matched  $n$ -grams in the testing data and the total occurrences of the matched  $n$ -grams in the training data

Short  $n$ -grams in the testing data are more easily to be found in the training set, and they occur much more frequently too. An extreme case is the high-frequency Chinese word "de" which occurs in almost every testing and training sentence. It is obvious that we do not need to retrieve the sentence ID and find alignment for each of its occurrences in the training data. Two strategies are applied to reduce the number of sentence ID retrieving operations and the *ASP* alignment:

- *ASP* will only be used for long phrases (e.g.,  $n > 3$ ). Translations for short phrases are trained from the off-line models. The mixture alignment model results in equivalent translation qualities as the pure online alignment model.
- Instead of applying *ASP* for all the phrase occurrences in the training corpus, only align up to a fixed number (e.g., 100) of sentence pairs.

Table 8 shows the number of retrieving operations and the time needed for locating all the substrings of TIDES02 testing data in the FBIS.gb training set. By restraining the total occurrences to be used by *ASP*, total locating time was reduced from 369 seconds to 1.3 seconds. Further relying on the off-line translation model for short phrase alignment, and use online methods to align phrases at least 3 words long reduced the time from 1.3 seconds to 0.33 second.

# Occurrence Retrieved for Each $n$ -gram	$n \geq$	# Retrieving Operations ( $\times 10^6$ )	Time
All	1	1004.0	369.00
	2	7.1	2.45
	3	0.4	0.42
100	1	3.1	1.30
	2	0.93	0.56
	3	0.18	0.33

Table 8. Reduce the locating time by restraining the length of phrases and the occurrences to be used by *ASP*

Figure 7 illustrates the time and space complexities of four alignment strategies: off-line alignment model restricting the length of phrases, such as the *ISA* model (Zhang 2003); off-line alignment model with no restriction on the phrase length, such as the *HMM* phrase alignment model (Vogel 2003); the estimated case for pure online alignment model where all substring occurrences are searched for their alignment; and the mixture online/offline alignment model.

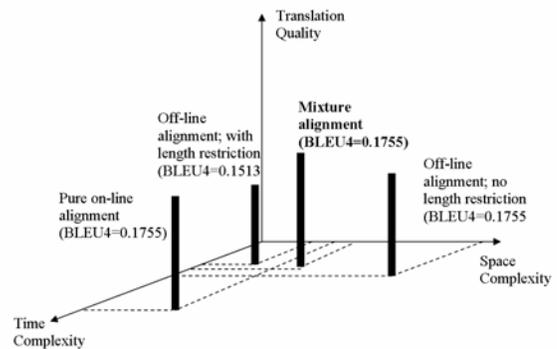


Figure 7. Time and space complexities of four alignment strategies

We have also developed a statistical machine translation system that is able to handle arbitrarily large bilingual corpora using the mixture alignment model. In this system, the decoder runs on one machine. It loads the language model and the off-line translation model for short phrases. Several other machines act as the bilingual corpus server, which return the alignments for long phrases in the testing sentences. The decoder then combines the

alignments from the off-line model for short phrases and the alignments from the bilingual corpus servers for long phrases and generates the translation hypothesis. By increasing the number of bilingual corpus servers, we can handle very large bilingual corpora.

## 6. Conclusion and Future Work

We presented a successful statistical machine translation system using the mixture online/offline alignment model. By allowing translating arbitrarily long phrases, the translation quality is significantly better. The fast substring search algorithm makes the *ASP* algorithm feasible in the online alignment scenario and the mixture alignment model makes the system efficient in both time and space complexity.

There are a number of possible extensions and refinements to the *ASP* alignment approach. One would be to calculate a constrained IBM4 alignment model. We will experiment with other word co-occurrence statistics, such as the mutual information, chi-square, or Dice coefficient.

So far the phrase alignment information is not used to update the word-to-word alignment probabilities. When using the IBM1 word alignment model a significant amount of the probability mass is distributed over word pairs, which are clearly no correct translation pairs. By updating the lexicon based on the phrase-to-phrase alignment the probability distribution could be focused more on the correct word pairs. This will be explored in the future.

## 7. References

- Rie Kubota Ando and Lillian Lee (2003). Mostly unsupervised statistical segmentation of Japanese kanji sequences. *Journal of Natural Language Engineering*, 9:127-149.
- Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. (1990). A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79-85.
- Philipp Koehn, Franz Josef Och, and Daniel Marcu.(2003). Statistical phrase-based translation. In *Proceedings of HLT/NAACL 2003*, Edmonton, Canada.
- Udi Manber and Gene Myers. (1990). Suffix arrays: a new method for on-line string searches. In *SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 319-327. Society for Industrial and Applied Mathematics.
- Udi Manber and Gene Myers.(1993). Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.*, 22(5):935-948.
- Daniel Marcu and WilliamWong. (2002). A phrase-based, joint probability model for statistical machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA.
- NIST. (2001). Automatic evaluation of machine translation quality using n-ram co-occurrence statistics. *Technical report, NIST*. Available at: <http://www.nist.gov/speech/tests/mt/>.
- Franz Josef Och, Christoph Tillmann, and Hermann Ney. (1999). Improved alignment models for statistical machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20-28, University of Maryland, College Park, MD, June.
- K. Papineni, S. Roukos, T. Ward, and W. Zhu. (2001). Bleu: a method for automatic evaluation of machine translation. *Technical Report RC22176(W0109-022)*, IBM Research Division, Thomas J. Watson Research Center.
- Stephan Vogel, Hermann Ney, and Christoph Tillmann. (1996). Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics*, pages 836-841. Association for Computational Linguistics.
- Stephan Vogel, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venogupal, Bing Zhao, and Alex Waibel. (2003). The CMU statistical translation system. In *Proceedings of MT Summit IX*, New Orleans, LA, September.
- Mikio Yamamoto and Kenneth W. Church. (2001). Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Comput. Linguist.*, 27(1):1-30.
- Ying Zhang, Stephan Vogel, and Alex Waibel. 2003. Integrated phrase segmentation and alignment algorithm for statistical machine translation. In *Proceedings of International Conference on*

*Natural Language Processing and Knowledge Engineering (NLP-KE'03)*, Beijing, China, October.

Ying Zhang, Stephan Vogel, and Alex Waibel (2004). Interpreting Bleu/NIST scores: How much improvement do we need to have a better system? In *Proceedings of LREC 2004*. Lisbon, Portugal.