Incorporating Constituent Structure Constraint into Discriminative Word Alignment

Wen-Han Chao
National Laboratory for Parallel
and Distributed Processing,
Changsha, China
cwh2k@163.com

Zhou-Jun Li
School of Computer Science & Engineering,
Beihang University,
Beijing, China
lizj@buaa.edu.cn

Abstract

We introduce an approach to incorporate the constituent structure constraint into a discriminative word alignment model by presenting the constituent constraint in an explicit way and using three operations to ensure the constraint when search the best word alignment. In this way, we will be able to make use of the weak order constraint induced by the inversion transduction grammars (ITG), as well as the flexibility of the discriminative word alignment framework to incorporating any other useful features.

1 Introduction

Most of recent statistical machine translation systems are based on word alignment, in which word re-ordering and multi-word alignment are two major problems. Most of initial work is derived from IBM models, or HMM model and Model 6 (Brown et al., 1993; Vogel et al. 1996; Och and Ney, 2003), which product good results on large sentence aligned bilingual corpora, especially when the two languages are closely related. But these generative models are complex, so that it is difficult to train the parameters in them, and incorporate new useful knowledge into them.

In order to restrict the word order further, especially considering the word order problem in two languages which are not closely related, many researchers introduce syntactic knowledge in word alignment, which mainly adopts a tree structure (Yamada and Knight, 2001; Cherry and Lin, 2003; Gildea 2004 etc.). In these models, a syntactic tree of one sentence is parsed which is used to constrain the words order in another sentence; or two trees are both parsed, and then the word alignment problem is to find a mapping between the nodes in the trees.

But in some languages, the syntactic trees are difficult to achieve, or the syntax between two languages are very different, so that the nodes in both trees are not easy to map. Accounting for these problems, Wu(1997) proposes a stochastic inversion transduction grammars(SITG), in which two simple operations are used to reorder the words, and in the end it will produce a binary tree. This model does not use traditional syntax, and the word alignment only needs to satisfy the constituent structure, so that it achieves a great flexibility while preserving a weak but effective word order constraint. But this model uses a dynamic programming algorithm to search the best word alignment, which complexity is $O(N^3V^3T^3)$, where V, T are the sentence lengths , when V and T are larger than 100 or even 50, it will be hard to bear in general PC machine. Besides, this model is also difficult to use other knowledge.

In order to incorporate various knowledge into the word alignment effectively, some researchers (Liu et al., 2005; Moore, 2005; Taskar et al., 2005) almost at the same time propose a discriminative word alignment framework, where the knowledge, such as POS and lexicon, are cast as some features. So that, when solving the word alignment, we only needs to select the features and train the corresponding weights of them. Their work shows that, when the features are selected appropriately, even some easy features will produce good results. Recently, the base feature in most of these models is the word correlation model, and then incorporating some other easy features, such as jump distance, POS etc.

In this paper, we propose a novel method, which will represent explicitly the constraints located in constituent structure produced by ITG. And then we can design more flexible search algorithm to find the best word alignment which is consistent with the constraints. To combine with other knowledge, we transfer the constituent structure constraint to a feature, which will be incorporated easily into a discriminative word alignment framework.

In the rest of this paper, we first introduce the constituent structure constraint in the ITG model in Section 2, and then describe how to transfer the constraint to four easy judgements in Section 3, and design a beam search algorithm to find the best word alignment which satisfies the constituent structure constraint in Section 4. We then present how to incorporate the constituent structure feature into a discriminative word alignment model in Section 5. Section 6 shows the results of the experiments, and we conclude in Section 7 and 8.

2 The Constituent Structure Constraint

Wu(1997) proposes a simple ITG which only consists of five types of rules:

$$A \xrightarrow{a} [A\ A]$$
$$A \xrightarrow{a} \langle A\ A \rangle$$
$$A \xrightarrow{b_{ij}} u_i/v_j$$
$$A \xrightarrow{b_{i\varepsilon}} u_i/\varepsilon$$
$$A \xrightarrow{b_{\varepsilon j}} \varepsilon/v_j$$

Where A is the only non-terminal symbol, and [] and <> represent the two operations which generate outputs in **straight** and **inverted** orientation respectively. u_i and v_j are terminal symbols, which represent the words in both languages, and ε represents the null word. The a , b_{ij} , $b_{i\varepsilon}$ and $b_{\varepsilon j}$ are the probabilities of productions. The last three rules are called lexical rules.

During the process of word alignment, this model will generate a binary branching tree, but the constituents are not the traditional ones, they only satisfy the following basic constraints:

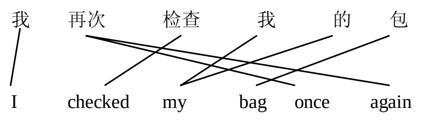
1) The words aligned are consecutive respectively.
2) There is a sub-tree for each constituent, it is a hierarchical structure, and the alignment is also hierarchical;
3) Each node in the sub-tree for the constituent *A* can not participate in another sub-tree for constituent *B*, unless *B* is an ancestor of *A*.

Although these constraints are easy, by using [] and <> operations, the model can almost interpret arbitrary word alignment, in which the word order in both sentence may vary greatly, as long as the word alignment satisfies the constituent structure constraints. So the model will be flexible to handle the words order during word alignment, especially when the two languages are very different in word order, such as Chinese and English. Figure 1 (a) and (b) illustrate a word-alignment and the corresponding constituent structure tree.

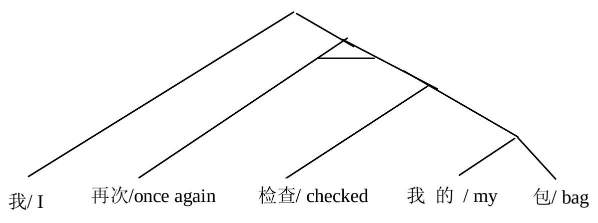
Besides, if the u_i and v_j are multi-words, the model will be able to produce many-to-many word alignment, but the multi-word must be consecutive.

Wu(1997) gives a dynamic programming algorithm for the ITG, which complexity is $O(N^3V^3T^3)$, which will slow down quickly when the sentence length becomes large. And how to incorporate effectively other useful knowledge into the model is another problem.

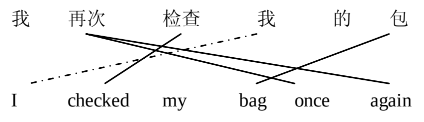
So we wish find a method, which can make use of the flexibility and effective word order constraint in the ITG while decreasing the complexity when searching word alignment at the same time, and can integrate easily the constituent structure constraint with other useful information.



(a) A valid word alignment example



(b) An ITG tree for the word alignment (a)



(c) An invalid word alignment example

Figure 1 A valid word alignment (a) and the corresponding ITG tree (b) where the line between the branches means an inverted orientation, otherwise a straight one, and an invalid alignment example, where the dot line is an invalid link (3-c) when given the other links.

3 Transforming the Constraint

In this paper, we use $<C,E>$ to represent a sentence pair, C and E are the sentences in both languages respectively, $C=c_1^m=c_1c_2....c_m$ and $E=e_1^n=e_1e_2....e_n$. We define a link $a(i,j;s,t)$ if the multi-words c_i^j and e_s^t are correspondences for each other, c_i^j and e_s^t may consist of zero, one or more words. If $i=j\neq 0$ or $s=t\neq 0$, they are single words respectively, and if one of the $i=j=0$ or $s=t=0$ exists, a is a null link. So, a word alignment A for the $<C,E>$ is a set of links: $A=\{a_1,a_2,...,a_n\}$. For the convenience, $a_{k-1}(i_{k-1},j_{k-1};s_{k-1},t_{k-1})$ and $a_k(i_k,j_k;s_k,t_k)$ refers that the a_k is behind the a_{k-1} in the sentence C, i.e. $i_k > j_{k-1}$. But it is not sure in the sentence E, i.e. s_k may be larger or smaller than t_{k-1}.

We also define a group $g=a_k^l=g(i_g,j_g;s_g,t_g)$, which is the combination of a_k, a_{k+1},… a_l, where $i_g...j_g$ and $s_g...t_g$ is consecutive and for any other $a_x(i_x,j_x;s_x,t_x)\in A$ and $x\notin\{k,l\}$, it must be $a_x\cap g=\phi$, i.e. $i_x,j_x\notin i_g...j_g$ and $s_x,t_x\notin s_g...t_g$. So a group is consecutive and integrated.

We can combine a group $g(i_g,j_g;s_g,t_g)$ with a link $a_x(i_x,j_x;s_x,t_x)$, $a_x\cap g=\phi$, to form a larger group $g'(i_{g'},j_{g'};s_{g'},t_{g'})$, in which $i_{g'}=\min(i_g,i_x)$, $j_{g'}=\max(j_g,j_x)$, $s_{g'}=\min(s_g,s_x)$, and $t_g'=\max(t_g,t_x)$. There may exist a link a_y, $a_y\cap g'\neq\phi$, but $a_y\not\subset g'$. To keep the integrality of the group g', we need to combine g' with the a_y, this process will continue until the formed group is integrated. The COMBINE procedure in Figure 3 shows the detail.

If $a_x\cap g=\phi$ and when combining a_x with g, it needs not add other link, i.e. the $g'(i_{g'},j_{g'};s_{g'},t_{g'})$ mentioned above is an integrated group, we say that the a_x is independent of g.

We define our word alignment problem as finding the best A that maximize $\Pr(A|C,E)$. And the process of forming a word alignment is: A is equal to ϕ at the beginning, and then add one a each time $A=A\cup\{a\}$, until no new a can be added anymore.

For each new added a_k, we will verify whether a_k is valid. If it is not valid, it can not be added to A.

If $A=\phi$, a_k is valid.

If $A\neq\phi$, there may exist at most two neighbors a_{k-1} and a_{k+1} which are near a_k and may form two groups g_{k-1} or g_{k+1} respectively. We say the new a_k is valid, when there exists at least one of groups g_{k-1} or g_{k+1}, which the new a_k is independent of , so that it may combine with the group to form an larger integrated group without involving other links.

There are four types of valid combinations as Figure 2 shows, in which a_k is independent of group g_{k-1} or g_{k+1}.

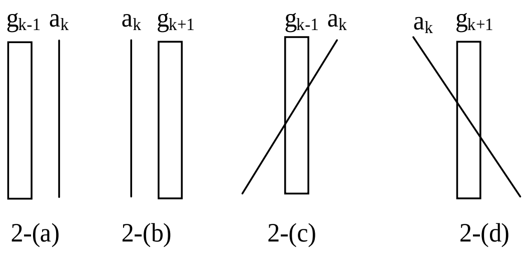


Figure 2 Four valid combination types

Where 2-(a) and 2-(b) represent that a_k is able to combine the g_{k-1} and g_{k+1} in a straight orientation; and 2-(c) and 2-(d) in an inverted orientation. Note, the groups may be any groups formed by a_{k-1} or a_{k+1}, i.e. their size is arbitrary, as long as the a_k is independent of them. If the new a_k accords with at least one of these types, we say it is valid.

Correspondingly, there are four types of invalid combinations as Figure 3 shows.

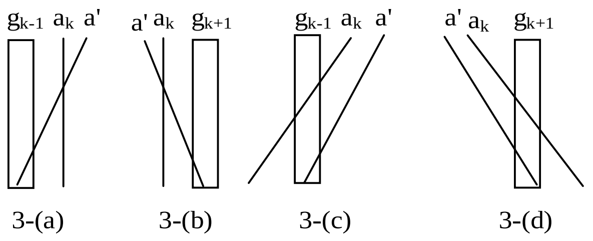


Figure 3 Four invalid combination types

Where the a' results in that the a_k can not be combined into groups.

In this way, our word alignment task is transferred to find a A in which each $a \in A$ accords with at least one type of the combinations in Figure 2.

Now, we will describe further how to verify the validity of the new a_k in a straight way. Observing that for each combination type in Figure 2, after combining the a_k and the group, the new group formed will take the a_k as the boundary. So, a simple way is to combine a_k with a_{k-1} or a_{k+1} directly, and verify whether the last formed integrated group exceeds the scope of a_k. If it exceeds, a_k does not accord with the combination, otherwise a_k is valid. For each type in Figure 1, the boundary of the new group that needs to be verified is different. For example, assuming the $a_k(i, j; s, t)$ and the last formed group $g'(i', j'; s', t')$, in 2-(a), we should verify whether $(j' \le j)$ and $(t' \le t)$; in 2-(c), we should verify whether $(j' \le j)$ and $(s \le s')$.

So, we achieve a simple method to verify if the new a_k is valid. Figure 4 gives the verifying procedure, which complexity is O(N), N is size of A, which is proportional to the lengths of the sentences.

Now, we define the process of word alignment more clearly as follows:

1. Initial $A = \phi$, and a candidate set M contains all the possible a.
2. We select one a from the M each time, and remove it from the M.
3. Verify whether the new a belongs to one of the four types in Figure 1. If TRUE, then $A = A \cup \{a\}$.
4. Iterate Step 2 and 3, until no other a can be added to A or $M = \phi$.

```
Procedure VERIFY_LINK_VALID
Input: $A$ and $a_k(i_k, j_k; s_k, t_k)$
Output: TRUE or FALSE
Begin
    If $A = \phi$, return TRUE ;
    Find the $a_{k-1}(i_{k-1}, j_{k-1}; s_{k-1}, t_{k-1})$ and $a_{k+1}(i_{k+1}, j_{k+1}; s_{k+1}, t_{k+1})$ ;
    If $a_{k-1}$ exists Then
      Initialize $g_{k-1} = a_{k-1}$;
      Combining the $g_{k-1}$ with $a_k$ to form a new group $g'(i', j'; s', t')$ ;
      if $(s_k > t_{k-1})$ then
          if $(j' \le j_k)$ and $(t' \le t_k)$ then return TRUE;
      if $(t_k < s_{k-1})$ then
          if $(j' \le j_k)$ and $(s_k \le s')$ then return TRUE;
    If $a_{k+1}$ exists Then
      Initialize $g_{k+1} = a_{k+1}$;
      Combining the $g_{k+1}$ with $a_k$ to form a new group $g'(i', j'; s', t')$ ;
      if $(s_{k+1} > t_k)$ then
          if $(i' \ge i_k)$ and $(s' \ge s_k)$ then return TRUE;
      if $(s_k > t_{k+1})$ then
          if $(i' \ge i_k)$ and $(t_k \ge t')$ then return TRUE;
    return FALSE;
End
Procedure COMBINE
Input: a group $g(i, j; s, t)$, a link $a_x(i_x, j_x; s_x, t_x)$ and $A$
Output: a new group $g'(i', j'; s', t')$
Begin
    $i' \leftarrow \min(i, i_x)$;
    $j' \leftarrow \max(j, j_x)$;
    $s' \leftarrow \min(s, s_x)$;
    $t' \leftarrow \max(t, t_x)$;
    $g' \leftarrow g'(i', j'; s', t')$;

    if $a_y(i_y, j_y; s_y, t_y) \in A$, $a_y \cap g' \ne \phi$ and $a_y \not\subset g'$ then

            return $g' \leftarrow$ COMBINE$(g', a_y, A)$

    return $g'(i', j'; s', t')$
End
```

Figure 4 Procedure to verify the validity of a_k

And we will prove that the word alignment produced through the above process will satisfy the constituent structure constraint in ITG.

Theorem 1

If each new added a accords with any of the four combination types in Figure 2, during the process word alignment, the word alignment produced in the end will satisfy the constituent structure constraint in ITG, i.e. it will form a constituent structure tree. And, any word alignment generated by the ITG will be able to be produced by the above process, i.e. adding a new a which belongs to one of the four combination types in the Figure 1 at each time.

Proof

1. $\Rightarrow$

At the beginning, $A = \phi$, so any new $a(i, j; s, t)$ will be valid, i.e. we can add a to A. Here we can add a new production to generate it:

$$A \xrightarrow{b_{ijst}} c_i^j / e_s^t .$$

where c_i^j and e_s^t are the multi-words in both languages, and they are generated according to probability b_{ijst}.

Assume that the A, formed previously by adding a, satisfies the constituent structure constraint, i.e., it forms a constituent tree. Now we will add a new a, if it is one of the four types in Figure 2, then there exists a group g in A, which is able to combine the a without involving the other links, i.e. $g' = g + a$. For g satisfies the constituent structure constraint, so there exists a production to generate g:

$$A \xrightarrow{bg} g ,$$

We can add three new productions to replace this production to generate g':

$$A \xrightarrow{a} [A\,A] \text{ or } A \xrightarrow{a} <A\,A>,$$

$$A \xrightarrow{bg} g$$

$$A \xrightarrow{ba} a$$

If g and a is combined in a straight orientation then we use $[]$ production and in an inverted orientation use $<>$.

In this way, the new $A = A + \{a\}$ generates a constituent tree and satisfies the constituent structure constraint. □

2. $\Leftarrow$

For a word alignment with constituent structure in ITG, we can get all its leaves firstly, and then generate one a for each leaf in the left to right order and add a to A. Initially, the $A = \phi$, so the first added a is valid.

Assume that all the previously added a belong to one of the four types of combinations in Figures 2. When adding a new a, the relationship between it and the previous a' may be:

1) They belong to a same parent node, i.e. they belong to a same constituent, and are the two branches of the constituent respectively. a' can form a group which only include a' and is independent of a. So they accord with the 2-a or 2-c combination type, and a is valid.
2) Their parent nodes are not same, but we can always find a node which are the same ancestor node of them, and it corresponds to the minimal constituent which includes both of the a and a'. At this time, a and a' belong to the two branches of the constituent respectively. So the branch where a' locates can form a group, which is independent of a. So a and the group accord with the 2-a or 2-c combination type, and a is valid.

So all word alignment generated by ITG can be produced by a process by adding a, which accords with one of the combination types in Figure 2. □

4 Alignment Search

In order to make use of the constituent structure constraint in the process of searching word alignment, we define the following three operations:

1. **Verify Operation**: verify operation is the most important operation, which will determine whether the new a is valid. Verify operation will pass the a through the procedure in Figure 4, to determinate whether the a belongs to one of the combination types in Figure 2.
2. **Insert Operation**: add the new a into A, i.e. $A = A + \{a\}$.
3. **Combine Operation**: combine the new added a with the adjacent group to form a larger group. During the process of word alignment, if $a(i, j; s, t)$ is adjacent with a group $g(u, v; x, y)$, we will merge them. The adjacency means the distance between a and g is zero in both sentences. There may be four cases corresponding with the four combination types. For example, in the first type, if $i = v + 1$ and $s = y + 1$, then we say that a is adjacent with g, and combine them to form a larger group $g'(u, j; x, t)$. And if two groups are adjacent, they can be merged into a larger group, too. After the word alignment, there may exist null links, so we need to combine groups which are not adjacent. At this time, each null link can be combined with any adjacent group. With the combine operation, we can output the constituent structure of the word alignment in the end.

By transferring the constituent structure constraint to the above three operations, we will be able to use any search algorithm to find the best word alignment. Here we design a fast beam search algorithm, which derived partially from the competitive linking algorithm (Melamed 2000), see Figure 5 for detail.

We collect firstly all possible a between the words in $< C, E >$ to form a candidate set M, and sort them by the $score(a)$, which may be the correlation probability of the words in a, and depends on the models we will used.

We then use the process defined in the Section 3 to produce a word alignment by using the Verify, Insert and Combine operations. For some different candidates may have an equal or near score, there may be many branches. But we will select the b branches to continue, and prune the branches by calculate the:

$$score(A) = \sum_a \log(score(a)) .$$

Our beam search algorithm has some characteristics: Firstly, it use the verify operation to determine whether the new a is valid, so the result word alignment is satisfied with the constituent structure constraint. Secondly, we use the combine operation when inserting new a, so the corresponding constituent tree of the word alignment will produced. Besides, our word alignment can be many-to-many, as long as the words are consecutive.

The complexity of this beam search algorithm is $O(bN^2)$, where b is the beam size, and N is the smaller sentence length in both sentences.

```
Algorithm 1
Input: sentence pair $<C,E>$
Output: a best word alignment of $<C,E>$
Begin
  $A=\phi$, $M$, $BWA=\{A\}$, $NB=\phi$, $ALIGNED=\phi$
  Do
    For each $A$ in $BWA$ Begin
      If ( $A$ has been all aligend) or (no other $a$ can be used in $M$ )
      Then $ALIGNED = ALIGNED + A$;
      Get the next N-best $a$ from $M$;
      For each $a$ Begin
        If( Execute the verify operation) Then
            $A = A \cup \{a\}$; Calculate $score(A)$; $NB = NB + A$;
      End
      Remove $A$ from $BWA$;
    End
    Sort all $A$ in $NB$
    Select the best $b$ branches and insert them into $BWA$;
    While ( $BWA$ is not NULL);
  Sort all $A$ in $ALIGNED$;
  Return the best $A$
End
```

Figure 5 The alignment search algorithm

5 Discriminative Word Alignment

Given a $<C,E>$, a discriminative word alignment is to find the best $A_{\max}$, so that:

$$A_{\max} = \arg\max_{A} \sum_{i=1}^{n} \lambda_i f_i(C,A,E)$$

Where the f_i represents the feature and λ_i is the corresponding weight of the feature. This framework makes it easy to combine various different knowledge in the word alignment problem, by representing the knowledge in the form of feature.

5.1 Features

In order to incorporate the constituent structure constraint into the discriminative word alignment model, there are two ways, the one is taking it as a search constraint, and using the verify operation to determine whether the new a is valid. Another way is to use a feature to represent this constraint. We use the latter one and our model will include the following features:

Constituent Structure Feature: count the number of the a which violates the constituent structure constraint. To ensure the result word alignment satisfies the cconstituent structure, we set a very small negative weight for this feature, so that the word alignment will not be used whenever this feature occurs.

Conditional Probability Model: we use a conditional probability as our base feature which accounts for the word correlation,

$$f_p(C,A,E) = \log P(A \mid C,E) = \sum \log p(a \mid c,e).$$

Where $p(a \mid c,e)$ is the alignment probability when c and e co-occur. There are various models, such as the Moore(2005),

$$p(a \mid c,e) = \frac{align(c,e) - d}{coocur(c,e)}.$$

Where $align(c,e)$ is the number of times that c and e are aligned, $coocur(c,e)$ is the number of times they co-occur, and d is a discount. Here we use the Φ^2 suggested by Gale and Church (1991), It is more robust because it considers all four cells in Table 1, i.e. it also considers the other alignments that c and e have participated in:

$$p(a \mid c,e) = \frac{(ad-bc)^2}{(a+b)(a+c)(b+d)(c+d)}.$$

	c	$\neg c$
e	$a = align(c,e)$	$b = align(\neg c,e)$
$\neg e$	$c = align(c,\neg e)$	$d = align(\neg c,\neg e)$

Table 1 An alignment contingency table (2*2)

Distortion Model: we count the jump distance for this model:

$$f_d(C,A,E) = \sum_i d_i.$$

Where the di represents the jump distance for each a, using one of the sentences as a reference. Our a is many-to-many, such as $a_k(i,j;s,t)$ and $a_{k-1}(i',j';s',t')$, if it is a straight orientation, the $di = s - t' - 1$; or $di = s' - t - 1$ in an inverted orientation.

Null word feature: we consider the effect for null alignment, by counting the number of null words in the word alignment.

5.2 Search

We use a similar beam search to find the best word alignment. But the $score(A)$ is calculated as follows:

$$score(A) = \sum_{i=1}^{4} \lambda_i f_i(C,A,E).$$

i.e., when adding a new a, we count the value for each feature, and multiply the weight respectively, and add them all to get the $score(A)$.

5.3 Training

The discriminative model needs to train the weights for the features, and we adopts the same **perception training** described in (Moore 2005) to train them, in which setting a initial set of the weights, and iterating through a small word-aligned development corpus for several times. At each time, it inputs each sentence pair in order, searches the best word alignment, and compares the features in the auto-generated word alignment with the reference word alignment. Then it updates the weights in the following way:

$$\lambda_i \leftarrow \lambda_i + \eta(f_i(C,A_{ref},E) - f_i(C,A_{auto},E)).$$

Where η is the learning rate. In this way, we will get the average weights over all the training data.

During our training process, we do not change the weights of the constituent structure feature and the conditional probability model, i.e., we only updates the weights of distortion model and null word feature.

Besides, the conditional probability model may be trained in various ways, but we use an iterate method as follows:

1. Initialize the conditional probability model using the Φ^2, where the $align(c,e) = coocur(c,e)$.
2. Train the weights of the current features.
3. Search the best word alignment for the training corpus.
4. Update the conditional probability model using the results in Step 3.
5. Iterate 2-4, until the model is converged or it meets the specified number for iterating.

6 Evaluation

We trained our method and model on a sentence-aligned bilingual Chinese and English corpus, which comprised 200,000 sentence pairs, and we also manually aligned 1,000 sentence pairs, in which 500 pairs of them were selected as development set, which were used to train the weights of our features, and the remained 500 pairs were used as test data. The manually aligned sentence pairs were aligned with constituent structure, i.e. they were aligned hierarchically. Table 2 shows the statistics of them.

		Chinese	English
Training Corpus	Sentences	200,000	
	Words	3,326,278	3,292,378
	Vocabulary	60,660	48,753
Develop Corpus	Sentences	500	
	Words	8,214	7,992
Test Corpus	Sentences	500	
	Words	8,066	7,997

Table 2 Statistics of training corpus, development corpus and test corpus

All the Chinese sentences are segmented using the ICTCLAS (zhang et al., 2003). And the English sentences are tokenized and stemmed, considering the Chinese word has no morphologic change and our bilingual corpus is relatively small. In this way the vocabulary size of English corpus decreases and the frequency of the word type increases.

We adopted the same evaluation methodology to the one in (Och and Ney 2003), i.e. recall, precision and alignment error rate (AER) to evaluate our model:

$$recall = \frac{|A \cap S|}{|S|},$$

$$precision = \frac{|A \cap P|}{|P|},$$

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

Where A is the set of alignments generated automatically by our word alignment system; S is the set of alignments which are manually marked "sure" , P is the set of alignments which are manually marked "possible" or "sure", then $S \subseteq P$

We prepared three experiments for comparison:

1. Training IBM-4 model on the training corpus using GIZA++ (Och and Ney 2003) and testing it.
2. Training a generative model derived directly from ITG, called HWA, where we consider only the conditional probability model
$$\log P(A \mid C, E) = \sum \log p(a \mid c, e).$$
We iterate to train the model as follows: Firstly, searching the best word alignment by Algorithm 1 in Figure 5, using:
$$score(A) = \log P(A \mid C, E) = \sum \log p(a \mid c, e).$$
In this way, the word alignment will satisfy the constituent structure constraint. Secondly, re-estimating the model using the best word alignments. If the model is converged or it meets the specified number for iterating, the training stops. And we then test this model using the same search algorithm.
3. Training the discriminative model in Section 5, called DWA. And we also train the weights in the development set. Finally, we test the model in the test set.

The results of the above three experiments are showed in Table 3.

Models	Precision	Recall	AER
HWA	0.823	0.769	0.203
DWA	0.858	0.766	0.188
IBM-4 $E \to C$	0.721	0.708	0.285
IBM-4 $C \to E$	0.772	0.702	0.263
IBM-4 union	0.704	0.719	0.288
IBM-4 intersection	0.787	0.738	0.237
IBM-4 refined	0.815	0.754	0.215

Table 3 Comparison of results for three models

Where $E \to C$ denotes treating the English as the source and Chinese as the target, and $C \to E$ is reverse. The intersection, union and refined denote three methods combining the alignments in both directions (i.e. $E \to C$ and $C \to E$) to improve the result (Och and Ney 2003).

As we can see, both HWA and DWA gives better results than IBM-4, and the AER of DWA reduces about 7% than HWA. But the recall of DWA and HWA is almost equal, while the precision of DWA increases about 4% than HWA. We conclude that the constituent structure constraint is very effective to restrict the word order, and by combining other knowledge with the constituent structure constraint, we will improve the result further.

7 Related Works

Many researchers have proposed models to make use of constituent structure or syntax to restrict the word order. But our constituent structure is derived from ITG (Wu 1997), which is not the traditional syntactic structure, so that it is very flexible to align two sentences with very different word order.

One of the main differences between our work and ITG is that we represent the constituent structure constraint explicitly, while it is implied in the rules of ITG. So that we can design some general search algorithms to decrease the complexity while keeping the constituent structure. In addition, we will be able to combine the constraint with other knowledge easily.

Zens et al. (2004) provides a decoder which satisfies the ITG constraint, their method is to ensure the fourth phrase can be translated when given the first two translated phrases, so it need to constrain the reordering of the third phrase. However, in our word alignment, the fourth phrase may be null aligned, so we just verify whether the current link is valid or not when given the preceding links. If it is invalid, it can not be added to the alignment, and the words may be aligned to null word.

Our discriminative word alignment model is similar with the other discriminative framework, except we design a new feature, which will incorporate more complex syntactic knowledge into the model.

8 Conclusions

We have presented a method to transfer the constituent structure constraint located in the ITG to four simple position judgement procedures in an explicit way. So, we can make use of the constituent structure to restrict the word order in word alignment easily. In this paper, we propose a beam search to decrease the complexity for find the best word alignment while preserve the constituent structure constraint. In addition, the word alignment we find is a many-to-many alignment, while the multi-words must be consecutive.

We also design a feature to represent the constituent structure constraint, so that we can incorporate it into the discriminative model.

The experiments show that the constituent structure is useful for word alignment, and the results will be improved by combining other features.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China under Grant Nos.60573057,60473057 and 90604007.

Bibliographical References

Peter F. Brown, Stepen A. Della Pietra, Vincent J. Della Pietra, and Rorbet. L.Mercer. (1993). The Mathematics of Statistical Machine Translation: Parameter estimation. Computational Linguistics, 19(2):263–312.

Colin Cherry and Dekang Lin. (2003). A Probability Model to Improve Word Alignment. In Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 88–95.

Colin Cherry and Dekang Lin. (2006). Soft Syntactic for Word Alignment through Discriminative Training. In Proceedings of COLING/ACL 2006. pp. 105.-112.

Willam. A. Gale and Kenneth W. Church. (1991). Identifying Word Correspondences in Parallel Texts. In 4th Speech and Natural Language Workshop, pages 152–157.

D. Gildea. (2004). Dependencies vs. constituents for treebased alignment. In Proceedings of the EMNLP, 214–221, Barcelona, Spain.

Yang Liu , Qun Liu and Shouxun Lin. (2005). Log-linear Models for Word Alignment In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, pp. 459–466.

I. Dan Melamed. (2000). Models of Translational Equivalence among Words. Computational Linguistics, 26(2):221–249.

R. Moore. (2005). A discriminative framework for bilingual word alignment. In Proceedings of HLT-EMNLP, pages 81–88, Vancouver, Canada, October.

Robert C. Moore, Wen-tau Yih and Andreas Bode. (2006). Improved Discriminative Bilingual Word Alignment. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 513–520, Sydney, July 2006.

Franz Joseph Och and Herman Ney. (2003). A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics, 29(1):19–52, March.

B. Taskar, S. Lacoste-Julien, and D. Klein. (2005). A discriminative matching approach to word alignment. In Proceedings of HLT-EMNLP, pages 73–80, Vancouver, Canada.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. (1996). HMM-based word alignment in statistical translation. In Proceedings of the COLING'96, pp. 836–841.

Dekai Wu. (1997). Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. Computational Linguistics, 23(3):374.

K. Yamada and K. Knight. (2001). A syntax-based statistical translation model. In Meeting of the Association for Computational Linguistics, pages 523–530.

R. Zens, H. Ney, T. Watanabe, and E. Sumita.（2004）, Reordering constraints for phrase-based statistical machine translation. In COLING'04: The 20th Int. Conf. on Computational Linguistics, pp. 205–211

Huaping Zhang, HongKui Yu, Deyi Xiong, and Qun Liu. (2003). HHMM-based Chinese lexical analyzer ICTCLAS. In proceedings of the second SigHan Workshop affiliated with 41th ACL, pp.184-187.

Bing Zhao and Stephan Vogel. (2003). Word Alignment Based on Bilingual Bracketing. In HLT-NAACL 2003 Workshop: Building and Using Parallel Texts Data Driven Machine Translation and Beyond, pp. 15-18.