Statistical Machine Translation Decoding Using Target Word Reordering

Jesús Tomás and Francisco Casacuberta

Institut Tecnològic d'Informàtica, Universidad Politècnica de Valencia
46071 Valencia, Spain
{jtomás, fcm}@upv.es

Abstract. In the field of pattern recognition, the design of an efficient decoding algorithm is critical for statistical machine translation. The most common statistical machine translation decoding algorithms use the concept of partial hypothesis. Typically, a partial hypothesis is composed by a subset of source positions, which indicates the words that have been translated in this hypothesis, and a prefix of the target sentence. Thus, the target sentence is generated from left to right obtaining source words in an arbitrary order. We present a new approach, where the source sentence is translated from left to right and the possible word reordering is performed at the target prefix. We implemented this approach using a multi-stack decoding technique for a phrase-based model, and compared it with both a conventional approach and a monotone approach. Our experiments show how the new approach can significantly reduce the search time without increasing the search errors.

1 Introduction

Statistical methods have proven to be valuable in tasks such as automatic speech recognition, and they present a new opportunity for automatic translation. The goal of statistical machine translation is to translate a given source language sentence $\mathbf{f} = f_1^{|\mathbf{f}|} = f_1 \dots f_{|\mathbf{f}|}$ to a target sentence $\mathbf{e} = e_1^{|\mathbf{e}|} = e_1 \dots e_{|\mathbf{e}|}$. The pattern recognition methodology most commonly used [2] is based on the definition of a function $\Pr(\mathbf{e}|\mathbf{f})$ that returns the probability of translating a given source sentence $\mathbf{f}$ into a target sentence $\mathbf{e}$. Once this function is estimated, the problem can be formulated as the search for a sentence $\mathbf{e}$ that maximizes the probability $\Pr(\mathbf{e}|\mathbf{f})$ for a given $\mathbf{f}$. Using Bayes' theorem, we can decompose the initial function in the language model and the inverse translation model:

$$\mathbf{e}' = \arg \max_{\mathbf{e}} (\Pr(\mathbf{e}) \Pr(\mathbf{f} | \mathbf{e})) \quad (1)$$

Nearly all statistical translation models try to establish the correspondence between source and target words by introducing the hidden variable of alignment [2]. Once the concept of alignment is formalized (see section 3 for a description), we introduce the variable of alignment, $\mathbf{a}$, into the previous equation in order to obtain the sum of all

possible alignments between $\mathbf{e}$ and $\mathbf{f}$. However, the search is usually performed using the so-called maximum approximation [8]:

$$\mathbf{e}' = \arg \max_{\mathbf{e}} \left(\Pr(\mathbf{e}) \sum_{\mathbf{a}} \Pr(\mathbf{f}, \mathbf{a} | \mathbf{e}) \right) \approx \arg \max_{\mathbf{e}} \left(\Pr(\mathbf{e}) \max_{\mathbf{a}} \Pr(\mathbf{f}, \mathbf{a} | \mathbf{e}) \right) \quad (2)$$

There are different approaches to define the concept of alignment. The most common approaches are the single-word-based alignment models. Models of this kind assume that an input word is generated by only one output word [1][2]. This assumption does not correspond to the nature of natural language.

One initiative for overcoming the above-mentioned restriction is known as the template-based approach [7]. In this approach, an entire group of adjacent words in the source sentence may be aligned with an entire group of adjacent target words. A template establishes the reordering between two sequences of word classes. However, the lexical model continues to be based on word-to-word correspondence.

Recently, a simple alternative to these models has been proposed, the phrase-based (PB) approach [5][10][13]. That is commented in the section 3.

2 Decoding Algorithms

In this section, we describe the most common statistical decoding algorithms. With the exception of the greedy algorithm [4], the rest of them use the concept of partial translation hypothesis to perform the search [1][8][12]. In a partial translation hypothesis, some of the source words have been used to generate a target prefix. Each hypothesis is scored according to the translation and language model. The most typical partial hypothesis comprises:

- $w \subset \{1..|\mathbf{f}|\}$: The coverage set indicates the positions of the source sentence that has been translated by this hypothesis.
- e_1^{lk} : The target prefix, where lk is the prefix length.
- g : The score of the partial hypothesis.

The translation procedure can be described as follows: The system maintains a large set of hypotheses, each of which has a corresponding translation score. This set starts with an initial empty hypothesis. The system selects one partial hypothesis to extend. The extension consists of selecting one or more untranslated words in the source sentence and also selecting one or more target words that are attached to the existing output prefix. In the new hypothesis, the source words are marked as translated and the probability cost of the hypothesis is updated. The extension of a partial hypothesis can generate hundreds of new partial hypotheses. The output of the search is the hypothesis that has the highest score and no untranslated source words (final hypothesis).

The strategies for selecting one partial hypothesis to be extended can be grouped into three types: If the best-first search is used, the A* algorithm is obtained; if the deep-first search is used, the multi-stack decoding algorithm is obtained. If the breadth-first search is used, the dynamic-programming algorithm is obtained.

2.1 A* Algorithm

The A* algorithm [4][8][12] is also known as the stack-decoding algorithm. All search hypotheses are managed in a priority queue (stack) ordered by their scores. This algorithm has one main drawback: short hypotheses have higher scores, therefore, the algorithm has a tendency to expand the short hypotheses first and the search is very slow. To solve this problem, we need to introduce the heuristic function [8][12]. This function estimates the probability of completing a partial hypothesis.

2.2 Multi-stack Decoding Algorithm

This algorithm, which was proposed in [1], tries to solve the problem of the previous algorithm by using a different approach. It uses a different stack to store the hypothesis depending on which words in the source sentence have been translated (the coverage set). Therefore, we may need up to a total of $2^{|I|}$ stacks. This procedure allows us to force the expansion of hypotheses with a different degree of completion. In each iteration, the algorithm covers all stacks with some hypotheses and extends the best one for each. After the first iteration, there is at least one final hypothesis. [1] proposes using the best final hypothesis, after each iteration, to establish a pruning criterion in order to erase the partial hypotheses that cannot improve the best final hypothesis.

2.3 Dynamic-Programming Algorithm

In this algorithm, the partial hypotheses are stored in the nodes of a search graph. This graph is explored in a breadth-first manner, that is, when a hypothesis is extended, all their predecessors¹ have also been extended.

The search space can be very large, so some authors make simplifying assumptions about the search space. [3] and [6] propose algorithms for IBM model 2, and [9] proposes algorithms for a monotone model. However, the most frequent strategy for making dynamic-programming feasible is beam search [9][13]. Beam search is based on histogram pruning, that is, in each node of graph, we extend only the H better hypotheses.

3 The Phrase-Based Model

In this approach the probability of a sequence of words in a source sentence being translated to another sequence of words in the target sentence is explicitly learnt. To define the PB model, we segment the source sentence $\mathbf{f}$ into K phrases ($\tilde{f}_i^K$) and the target sentence $\mathbf{e}$ into K phrases ($\tilde{e}_i^K$). A uniform probability distribution over all possible segmentation is assumed ($\alpha(\mathbf{e})$).

¹ h' is a predecessor of h if the words translated by h' are a subset of the words translated by h .

$$\Pr(\mathbf{f} | \mathbf{e}) = \alpha(\mathbf{e}) \sum_K \sum_{\tilde{\mathbf{e}}_1^K : \tilde{\mathbf{e}}_1^K = \mathbf{e}} \sum_{\tilde{f}_1^K : \tilde{f}_1^K = \mathbf{f}} \Pr(\tilde{f}_1^K | \tilde{\mathbf{e}}_1^K) \quad (3)$$

If we assume a monotone alignment, that is, the target phrase in position k is produced only by the source phrase in the same position [10], we can write:

$$\Pr(\tilde{f}_1^K | \tilde{\mathbf{e}}_1^K) = \prod_{k=1}^K p(\tilde{f}_k | \tilde{\mathbf{e}}_k) \quad (4)$$

Where the parameter $p(\tilde{f} | \tilde{\mathbf{e}})$ estimates the probability that the phrase, $\tilde{\mathbf{e}}$, be translated to the phrase $\tilde{f}$. These are the only parameters of this model. A phrase can be comprised by a single word. Thus, the conventional, word-to-word statistical dictionary is included.

If we permit the reordering of the target phrases, a hidden phrase level alignment variable, $\tilde{\mathbf{a}}$, is introduced. In this case, we assume that the target phrase in position k is produced only by the source phrase in position $\tilde{a}_k$.

$$\Pr(\tilde{\mathbf{e}}_1^K | \tilde{f}_1^K) = \sum_{\tilde{\mathbf{a}}} \prod_{k=1}^K p(\tilde{a}_k | \tilde{a}_1^{k-1}) p(\tilde{\mathbf{e}}_k | \tilde{f}_{\tilde{a}_k}) \quad (5)$$

For the distortion model, we assume a first-order alignment that depends only on the distance of the two phrases [7]:

4 Monotone Algorithm

In this section, we present a search algorithm for the monotone PB model (equation 4) based on multi-stack decoding. This algorithm is similar to the one in [1] but takes advantage of the sequentiality of the model. We define a hypothesis search as the triple (mk, e_1^{lk}, g) , where mk is the length of the source prefix we are translating in the hypothesis (that is f_1^{mk}). The sequence of lk words e_1^{lk} is the target prefix that has been generated. And g is the score of the hypothesis.

Following the multi-stack decoding approach, hypotheses are stored in different priority queues according to their value of mk . This results in the following algorithm:

```

Create priority queues from  $Q_0$  to  $Q_{|f|}$ 
Initialize  $Q_0$  with the empty hypothesis ( $mk=0, lk=0, g=1$ )
Repeat max_expan times or until no more hypotheses to extend
  For each queue from  $Q_0$  to  $Q_{|f|-1}$ :
    Pop the hypothesis with the highest score;  $h=(mk, e_1^{lk}, g)$ 
    For  $|\tilde{f}|=1$  to  $|f|-mk$ 
       $\tilde{f} = f_{mk+1}^{mk+|\tilde{f}|}$ 
      For each  $\tilde{\mathbf{e}}$  with  $p(\tilde{f} | \tilde{\mathbf{e}}) > 0$ 
        Push in  $Q_{mk+|\tilde{f}|}$  the hypothesis:
           $(mk+|\tilde{f}|, e_1^{lk} \tilde{\mathbf{e}}, g + \Pr(\tilde{\mathbf{e}} | e_1^{lk}) \cdot p(\tilde{f} | \tilde{\mathbf{e}}))$ 
The hypothesis of  $Q_{|f|}$  with the highest score is the output

```

Fig. 1. Multi-stack decoding monotone algorithm for PB model.

The hypothesis extension consists of: selecting a phrase $\tilde{f}$ from the source sentence, starting with the last word translated and with any length; selecting $\tilde{e}$, a possible translation of $\tilde{f}$; and appending it to the target prefix.

The parameter *max_expan* limits the number of iterations of the algorithm. A typical value used is 10; results do not improve with greater values. The introduction of this parameter permits a very fast search. We can translate several hundred words in a second.

To improve the translation speed, pruning criteria is introduced to erase hypotheses with a low probability of becoming the best output. [1] proposes using a threshold by stack, which is calculated from the best final hypothesis up to that point. In our implementation, we use a very simple solution. If a hypothesis has a score that is higher than the best output, the hypothesis is discarded. The same pruning criterion is used in the two algorithms described below.

5 Source Word Reordering Algorithm

The monotone algorithm does not permit the reordering of the target phrases. In this section, we present a search algorithm for the nonmonotone model (equation 5). This algorithm is similar to the one described in [1].

We define a hypothesis search as the triple (w, e_1^{lk}, g) , where $w \subset \{1..|f|\}$ is the coverage set that defines which positions of source words have been translated; e_1^{lk} is the target prefix that has been generated; and g is the score of the hypothesis. A similar definition of a hypothesis search can be found in [1][12][13].

For a better comparison of hypotheses, [1] proposes storing each hypothesis in different priority queues according to their value of w . The number of possible queues can be very high ($2^{|f|}$); thus, the queues are created on demand. Here is the algorithm:

```

Initialize  $Q_{\emptyset}$  with the null hypothesis ( $w=\emptyset, lk=0, g=1$ )
Repeat max_expan times or until no more hypotheses to expand
  For each queue in order of index cardinality
    Pop the highest scored hypothesis;  $h=(w, e_1^{lk}, g)$ 
    For  $|\tilde{f}|=1$  to  $|f|$ 
      Let  $j_{init} = \min_{n=1..|f|} (n \notin w)$ 
      For  $j=j_{init}$  to  $j_{init}+win\_search-1$  with  $w \cap \{j, \dots, j+|\tilde{f}|-1\} = \emptyset$ 
         $\tilde{f} = f_{j:j+|\tilde{f}|-1}$ 
        For each  $\tilde{e}$  with  $p(\tilde{f}|\tilde{e}) > 0$ 
          Push in  $Q_{w \cup \{j, \dots, j+|\tilde{f}|-1\}}$  the hypothesis:
             $(w \cup \{j, \dots, j+|\tilde{f}|-1\}, e_1^{lk} \tilde{e}, g \cdot \Pr(\tilde{e} | e_1^{lk}) \cdot p(\tilde{f} | \tilde{e}))$ 
The hypothesis of  $Q_{\{1..|f|\}}$  with the highest score is the output

```

Fig. 2. Multi-stack decoding source word reordering algorithm for PB model.

The principal difference of this algorithm with the monotone algorithm is the hypothesis extension. Now, the source phrase $\tilde{f}$ can be selected starting at any position j , with the restriction of the words of $\tilde{f}$ which have not been translated in the hypothesis to be extended ($w \cap \{j, \dots, j+|\tilde{f}|-1\} = \emptyset$). This algorithm uses an additional restriction proposed in [1]. The initial source position j is selected only through the *win_search* positions, beginning at the first nontranslated word.

This algorithm is much slower than the monotone algorithm. First, it introduces an additional order of magnitude selecting the position of the input phrase. Second, it extends a hypothesis from each stack created. Afterwards, there are $|f|$ stacks and there can be now reach up to $2^{|f|}$.

6 Target Word Reordering Algorithm

In order to improve the search time, we propose an algorithm that has a structure that is similar to the monotone algorithm. We take the words of f from left to right, and we introduce a possible word reordering at the output prefix.

Similar to the monotone algorithm, we define a hypothesis search as the triple (mk, e_1^{lk}, g) . This hypothesis indicates that the source prefix f_1^{mk} has been translated by the target prefix e_1^{lk} with a score of g . Different to the previous case, we can introduce the special token $\langle \text{nul} \rangle$ at the output prefix. The meaning of this token is that in a future expansion, the token $\langle \text{nul} \rangle$ must be replaced by a sequence of words. See Figure 3 for an example. In principle, a hypothesis could include an arbitrary number of $\langle \text{nul} \rangle$ tokens; however, in our implementation, we allow only one. Therefore, we can distinguish between two classes of hypotheses. A hypothesis is closed if it does not contain the token $\langle \text{nul} \rangle$, and it is open if it contains this token. In the latter case, if $\langle \text{nul} \rangle$ token is at position i , we can represent the target prefix as $e_1^{i-1} \langle \text{nul} \rangle e_{i+1}^{lk}$.

$$(0, "", 1) \rightarrow (1, "el", g_1) \rightarrow (2, "el \langle \text{nul} \rangle de configuración", g_2) \rightarrow (3, "el programa de configuración", g_3)$$

Fig. 3. Example of a sequence of hypotheses that permits translating the Spanish sentence ‘*el programa de configuración*’ from the English sentence ‘*the configuration program*’.

The hypothesis extension begins selecting $\tilde{f}$, starting at the position $mk+1$ of f . Then, $\tilde{e}$ is selected as a possible translation of $\tilde{f}$. If the hypothesis to be extended is closed, two new hypotheses are created; one is created by putting $\tilde{e}$ at the right of the target prefix, and the other is created by putting $\langle \text{nul} \rangle \tilde{e}$ at the right of the target prefix. If the hypothesis is open, four new hypotheses are created; one that closes the hypothesis, replacing the token $\langle \text{nul} \rangle$ by $\tilde{e}$, and three that keep the new hypotheses open, putting $\tilde{e}$ at the left or right of $\langle \text{nul} \rangle$, or putting $\tilde{e}$ at right of the target prefix.

This algorithm uses a different approach to calculate the distortion probability. A similar approach for the previous algorithm is not possible. When we open a hypothesis, we do not know the final position that a phrase will take. Thus, we have a different parameter distortion for each type of extension. If the hypothesis is closed, we use de probabilities p_m to keep it closed and $1-p_m$ to open it. If the hypothesis is open, we have four different extension types with the probabilities p_c, p_i, p_d and $1-p_c-p_i-p_d$.

Another problem we need to solve is the evaluation of the language model in the partial hypotheses. In the two other algorithms, the target prefix is created from left to right, so we can easily use an n-gram model. In this algorithm, if we have an open hypothesis, we cannot calculate the language model contribution of the right part of the prefix after the $\langle \text{nul} \rangle$ token, since we do not know which words will be replaced by $\langle \text{nul} \rangle$. To solve this problem, we compute an estimation of the language model contribution. It consists of assigning the probability of its unigram to the word at the right of $\langle \text{nul} \rangle$. We assign the probability of the bigram to the next word, etc. When a hypothesis is closed, this estimation is replaced by the true language model contribution.

```

Create priority queues from  $Q_0$  to  $Q_{|f|}$ 
Initialize  $Q_0$  with the null hypothesis ( $mk=0, lk=0, g=1$ )
Repeat max_expan times or until no more hypotheses to extend
  For each queue from  $Q_0$  to  $Q_{|f|-1}$ 
    Pop the hypothesis with the highest score;  $h=(mk, e_1^{lk}, g)$ 
    For  $|\tilde{f}|=1$  to  $|f|-mk$ 
       $\tilde{f} = f_{mk+1}^{\tilde{f}}$ 
      For each  $\tilde{e}$  with  $p(\tilde{f}|\tilde{e}) > 0$ 
        Push in  $Q_{mk+|\tilde{f}|}$  the hypotheses:
        if  $h$  is closed:
          ( $mk+|\tilde{f}|, e_1^{lk} \tilde{e}, g \cdot \text{Pr}(\tilde{e} | e_1^{lk}) \cdot p(\tilde{f}|\tilde{e}) \cdot p_m$ )
          ( $mk+|\tilde{f}|, e_1^{lk} \langle \text{nul} \rangle \tilde{e}, g \cdot \text{Pr}(\tilde{e} | \langle \text{nul} \rangle) \cdot p(\tilde{f}|\tilde{e}) (1-p_m)$ )
        if  $h$  is open:
          ( $mk+|\tilde{f}|, e_1^{lk} \tilde{e}, g \cdot \text{Pr}(\tilde{e} | \langle \text{nul} \rangle e_{i+1}^{lk}) \cdot p(\tilde{f}|\tilde{e}) (1-p_c-p_i-p_d)$ )
          ( $mk+|\tilde{f}|, e_1^{i-1} \tilde{e} e_{i+1}^{lk}, g \cdot \text{Pr}(\tilde{e} | e_{i+1}^{lk} | e_1^{i-1}) \cdot p(\tilde{f}|\tilde{e}) / \text{Pr}(e_{i+1}^{lk} | \langle \text{nul} \rangle) \cdot p_c$ )
          ( $mk+|\tilde{f}|, e_1^{i-1} \tilde{e} \langle \text{nul} \rangle e_{i+1}^{lk}, g \cdot \text{Pr}(\tilde{e} | e_1^{i-1}) \cdot p(\tilde{f}|\tilde{e}) \cdot p_i$ )
          ( $mk+|\tilde{f}|, e_1^{i-1} \tilde{e} \langle \text{nul} \rangle \tilde{e} e_{i+1}^{lk}, g \cdot \text{Pr}(\tilde{e} | e_{i+1}^{lk} | \langle \text{nul} \rangle) \cdot p(\tilde{f}|\tilde{e}) / \text{Pr}(e_{i+1}^{lk} | \langle \text{nul} \rangle) \cdot p_d$ )
    Closed hypothesis of  $Q_{|f|}$  with highest score is the output

```

Fig. 4. Multi-stack decoding target word reordering algorithm for the PB model.

7 Experimental Results

In order to compare the three decoding algorithms introduced above, several experiments were carried out. We use the Hansards task which consists of proceedings of the Canadian parliament. The training corpus consists of 50,000 sentences (2.1 million French and 1.9 million English words). The vocabulary size is 29,479 for French

and 37,554 for English. With this corpus we train a phrase-based model [11] using a maximum length of phrase of 4 words. We obtain 916,414 parameters in this model. We used a test comprised of 240 bilingual sentences, uniformly distributed across the source lengths 5, 10, 15, ..., 60. We use two criteria to evaluate the algorithms, the translation accuracy, measured in word error rate, (WER)[8][9] and the translation speed, measured in words per second.

Table 1 shows the effect *max_expand* parameter in the three algorithms. Monotone algorithm obtains a WER a little worst than nonmonotone algorithms. In other tasks, such as a reduced domain task or a task between Romanic languages, it obtains the same results as nonmonotone algorithms. With respect the search speed, the monotone algorithm is very fast. It obtains the best results with a value of *max_expand*=8. For this value, it can translate 37 words per second. Source word reordering (SWR) algorithm obtains somewhat better results than monotone algorithm. To obtain this improvement, it needs a high value of *max_expand* and it is very slow. We cannot use this algorithm in real time tasks. Target word reordering (TWR) algorithm obtains similar results to the SWR algorithm. However, it is faster. It obtain the best results with a value of *max_expand*=32. For this value, it can translate 4.2 words per second.

Table 1. Effect of *max_expand* parameter on WER and translation speed for the three decoders (*win_search*=6).

<i>max_expand</i>		1	2	4	8	16	32	64
monotone	WER	74.49	74.54	74.46	74.46	74.40	74.39	74.40
	word/sec.	268	142	73.0	37.2	18.8	8.5	3.1
source word reordering	WER	74.60	74.45	74.41	74.35	74.38	74.13	-
	word/sec.	2.2	1.0	0.49	0.23	0.11	0.05	-
target word reordering	WER	74.49	74.54	74.39	74.40	74.31	74.12	74.14
	word/sec.	160	80.3	39.3	19.3	8.6	4.2	2.2

Table 2 shows the effect of sentence length on the three decoding algorithms. Usually, sort sentences are better translated than long sentences; however, the difference is small. Except for sentences with 5 words, the translation speed is independent of the sentence length in monotone and TWR algorithms. However, in SWR algorithm, the translation speed decreases with the sentence length in sentences with less than 25 words.

Table 2. Effect of sentence length on WER and translation speed for the three decoders (*max_expand*=20, *win_search*=6).

Sentence length		5	10	15	20	25	30	35	40	45	50	55	60
monotone	WER	63.0	69.5	79.9	75.2	79.8	76.2	71.9	74.9	72.0	69.9	74.9	75.8
	word/sec.	31.7	13.9	15.5	14.0	15.7	12.8	14.3	14.7	14.5	14.5	14.5	14.3
source word reordering	WER	63.0	73.1	78.7	72.8	79.1	75.0	71.7	74.8	71.2	69.3	73.9	74.5
	word/sec.	1.2	0.31	0.17	0.12	0.14	0.11	0.12	0.13	0.12	0.12	0.11	0.13
target word reordering	WER	62.0	68.5	79.0	74.0	79.2	75.0	71.9	74.6	71.0	69.3	73.6	74.9
	word/sec.	11.2	6.4	7.5	6.8	7.7	6.4	7.2	7.4	7.8	7.5	7.2	7.5

8 Conclusions

In this paper, we have described three search algorithms based on multi-stack decoding techniques. We have used them with the phrase-based model [10][13], and we think these algorithms can be easily adapted to other models like template-based [7] or IBM models [1][2].

In our experiments, monotone search obtained results in WER that were comparable to non monotone search. However, it is much faster. We obtain similar results in other tasks and in other models (like template-based) which are not reported in this paper [11]. For nonmonotone search, we have proposed a new approach, where the source sentence is translated from left to right and the word reordering is computed at the target prefix. We have compared it with the conventional approach. The experiments show that for similar search results the new approach is faster and requires less memory for stacks.

In the future, we are interested in applying the dynamic-programming technique to these algorithms, and comparing them with the stack-decoding technique. We are also interested in using these algorithms with other models, such as the IBM models.

Acknowledgements

This work has been partially supported by the Spanish project TIC2003-08681-C02-02 and the IST Program of the European Union under grant IST-2001-32091.

References

1. Berger, A.L., Brown, P.F., Della Pietra, S.A., Della Pietra, V.J., Gillett, J.R., Lafferty, J.D., Mercer, R.L., Printz, H., Ures, L.: Language Translation Apparatus and Method Using Context-Based Translation Models. United States Patent No. 5,510,981 (1996)
2. Brown, P.F., Della Pietra, S.A., Della Pietra, V.J., Mercer, R.L.: The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, vol. 19 no. 2, (1993) 263–311
3. Garcia-Varea, I., Casacuberta, F.: An Iterative, DP-based Search Algorithm for Statistical Machine Translation. *Proceedings of the ICSLP-98*, Sidney, Australia (1998)
4. Germann, U., Jahr, M., Knight, K., Marcu, D. Yamada, K: Fast Decoding and Optimal Decoding for Machine Translation. *Proceedings of ACL 2001*. Toulouse, France (2001)
5. Marcu, D., Wong W.: A Phrase-Based, Joint Probability Model for Statistical Machine Translation. *Proc. of the Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA, July (2002)
6. Niessen, S., Vogel, S., Ney, H., Tillmann, C.: A DP based Search Algorithm for Statistical Machine Translation. *Proceedings of 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, Montreal, Canada (1998)

7. Och, F.J., Ney, H.: Improved Statistical Alignment Models. Proc. of the 38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, China, October (2000)
8. Och, F.J., Ueffing, N., Ney, H.: An Efficient A* Search Algorithm for Statistical Machine Translation. Proc. of Data-Driven Machine Translation Workshop, Toulouse, France (2001)
9. Tillmann, C., Vogel, S., Ney, H.: A DP based Search Using Monotone Alignments in Statistical Translation. Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, Madrid, Spain (1997)
10. Tomás, J., Casacuberta, F.: Monotone Statistical Translation using Word Groups. Proceedings of the Machine Translation Summit VIII, Santiago de Compostela, Spain (2001)
11. Tomás, J., Casacuberta, F.: Combining Phrase-Based and Template-Based alignment models in Statistical Translation. Lecture Notes in Computer Science 2652, (2003)
12. Wang, Y.Y., Waibel, A.: Fast Decoding for Statistical Machine Translation. Proceedings of the International Conference on Speech Language Processing, (1998) 2775-2778
13. Zens, R., Och, F.J., Ney, H.: Phrase-Based Statistical Machine Translation. Proceedings of the Conference on Empirical Methods for Natural Language Processing (2002)