SEARCH ALGORITHMS FOR STATISTICAL MACHINE TRANSLATION BASED ON DYNAMIC PROGRAMMING AND PRUNING TECHNIQUES

Ismael García-Varea[1], **Francisco Casacuberta**[2]

[1] Departamento de Informática
Universidad de Castilla-La Mancha
02071 Albacete, SPAIN
ivarea@info-ab.uclm.es
[2]Instituto Tecnológico de Informática
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
46071 Valencia, SPAIN
fcn@iti.upv.es

Abstract

The increasing interest in the statistical approach to Machine Translation is due to the development of effective algorithms for training the probabilistic models proposed so far. However, one of the open problems with statistical machine translation is the design of efficient algorithms for translating a given input string. For some interesting models, only (good) approximate solutions can be found. Recently, a dynamic programming-like algorithm for the IBM-Model 2 has been proposed which is based on an iterative process of refinement solutions. A new dynamic programming-like algorithm is proposed here to deal with more complex IBM models (models 3 to 5). The computational cost of the algorithm is reduced by using an alignment-based pruning technique. Experimental results with the so-called "Tourist Task" are also presented.

Keywords

Statistical Machine Translation, Search Algorithms, Pruning Techniques

Introduction

The goal of the translation process of *statistical machine translation (SMT)* can be formulated as follows: given a source language string $f_1^J = f_1 \dots f_J$, search for the target language string $\hat{e}_1^I = \hat{e}_1 \dots \hat{e}_I$ that maximize

$$\hat{e}_1^I = \underset{e_1^I}{\operatorname{argmax}}\, Pr(e_1^I | f_1^J). \quad (1)$$

In our case, the source language is Spanish and the target language is English. According to Bayes' decision rule, we have to choose the target string that maximizes the product of the target language model $Pr(e_1^I)$ and the string translation model $Pr(f_1^J|e_1^I)$. The equation that models this process is:

$$\hat{e}_1^I = \underset{e_1^I}{\operatorname{argmax}}\{Pr(e_1^I) \cdot Pr(f_1^J|e_1^I)\} \quad (2)$$

Many existing systems for SMT (Niessen et al., 1998; Wang and Waibel, 1997) make use of a special way of structuring the string translation model as proposed by (Brown et al., 1993): The correspondence between the words in the source and the target string is described by alignments that assign one target word position to each source word position.

One of the open problems with SMT is the design of efficient algorithms for searching the translation of a given input string, that is to solve the global search maximization problem.

Interesting translation models were proposed in (Brown et al., 1993). However, the corresponding search algorithms for the IBM-Model 1 to IBM-Model 5 in (Brown et al., 1993) are based on a certain type of the $A^\star$ algorithm (Brown et al., 1990) or on the *stack decoding* approach (Al-Onaizan et al., 1999; Wang and Waibel, 1997).

There are also dynamic-programming-based search algorithms such as (Tillmann et al., 1997a; Tillmann et al., 1997b) for the first-order models proposed in (Vogel et al., 1996) and the algorithm proposed in (Tillmann, 2001) for the IBM-Model 4 which is inspired on the Traveler Salesman problem. A dynamic-programming search algorithm for the IBM-Model 2 is also proposed in (García-Varea et al., 1998).

In this paper we present how to deal with the most complex IBM models (IBM-Model 3 to IBM-Model 5) using the algorithm proposed in (García-Varea et al., 1998) and an alignment-based pruning technique in order to reduce the high computational cost.

For better understanding, in the following section we review the search algorithm proposed in (García-Varea et al., 1998) and we present an intuitive idea about how the search process is performed. In the next section, we explain how to deal with the most complex models proposed in (Brown et al., 1993) and the pruning techniques used. Afterwards, we present the experimental results we have carried out with the so-called "Tourist Task" (Amengual et al., 1996). The last section presents the main conclusions of this work and possible future work to be done in this direction.

A review of the iterative DP-based search

The iterative DP-based search algorithm (IDPS) proposed in (García-Varea et al., 1998) is a heuristic dynamic-programming search algorithm which uses the IBM-Model 2 translation model (Brown et al., 1993) and is based on an iterative process of refinement solutions.

The translation models introduced by (Brown et al., 1993) are based on the concept of alignment between the components of the *translation pairs* (f_1^J, e_1^I).

Formally, an alignment is a mapping between the sets of positions in f_1^J and e_1^I: $a \subset 1 \ldots J \times 1 \ldots I$. However, in (Brown et al., 1993), the concept of alignment is restricted to being a function $a : 1 \ldots J \rightarrow 0 \ldots I$, where $a_j = 0$ means that the position j in f_1^J is not aligned with any position of e_1^I (or aligned to the *null word* e_0). All the possible alignments between e_1^I and f_1^J are denoted by $\mathcal{A}(f_1^J, e_1^I)$ and the probability of translating a given e_1^I into f_1^J by an alignment is denoted by $Pr(f_1^J, a_1^J | e_1^I)$. Therefore,

$$Pr(f_1^J | e_1^I) = \sum_{a \in \mathcal{A}(f_1^J, e_1^I)} Pr(f_1^J, a_1^J | e_1^I) \qquad (3)$$

The IBM-Model 2 for $Pr(f_1^J, a_1^J | e_1^I)$ proposed in (Brown et al., 1993) is

$$Pr_{M2}(f_1^J, a_1^J | e_1^I) = \prod_{j=1}^{J} t(f_j | e_{a_j}) \cdot a(a_j | j, J, I) \qquad (4)$$

where $t(f_j | e_{a_j})$ is the *translation probability* of the input word f_j given the output word e_i, and $a(a_j | j, J, I)$ is the *alignment probability*. This distribution gives us the alignment probability of the i-th word in the target sentence, given any position in the source sentence and the length of both sentences. If equation (4) is used in (3), we have,

$$Pr_{M2}(f_1^J | e_1^I) = \prod_{j=1}^{J} \sum_{i=0}^{I} t(f_j | e_i) \cdot a(i | j, J, I) \qquad (5)$$

The experiments carried out in (García-Varea et al., 1998) were performed using a n-gram language model as the general language model used in (2). For the sake of simplicity in the notation, the IDPS algorithm is outlined here using a bigram language model $B = (p_b)$.

Given an input string e_1^I and an IBM-Model 2 translation model, and supposing that the length of the output string I is known, the score associated to equation (2) using equation (5) for a hypothesis translation e_1^I can be computed as:

$$\max_{e_1^I} \left(Pr_B(e_1^I) \cdot Pr(f_1^J | e_1^I) \right) = \max_{e_1^I} \left(\prod_{i=1}^{I} p_b(e_i | e_{i-1}) \prod_{j=1}^{J} \sum_{i=0}^{I} t(f_j | e_i) \cdot a(i | j, |J, I) \right). \qquad (6)$$

This maximization can be rewritten as:

$$\max_{e_1^I} \left(Pr_B(e_1^I) Pr_{M2}(f_1^J | e_1^I) \right) = \max_{e_1^I} \left(T(e_I, I) \prod_{j=1}^{J} Q(e_I, I, j) \right) \qquad (7)$$

Where, in general: for $1 < i < I$ and for all possible target word e

$$\begin{aligned} T(e,i) &= p_b(e|\hat{e}(e,i)) \cdot T(\hat{e}(e,i), i-1) \\ Q(e,i,j) &= Q(\hat{e}(e,i), i-1, j) + t(f_j|e) \cdot a(i|j,J,I) \\ \hat{e}(e,i) &= \operatorname*{argmax}_{e'} \Big(p_b(e|e') \cdot T(e', i-1) \times \\ &\quad \prod_{j=1}^{J} \left[Q(e', i-1, j) + t(f_j|e) \cdot a(i|j,J,I) \right. \\ &\quad \left. + R(j, i+1) \right] \Big), \end{aligned} \qquad (8)$$

with

$$R(j,i) = \sum_{k=i}^{I} t(f_j | \bar{e}_k) \cdot a(k | j, J, I), \qquad (9)$$

where $\bar{e}_1^J$ is a guessed output. Note that $\bar{e}_k$ $(i+1 \leq k \leq I)$ corresponds to output symbols that are not yet explored. Therefore, $\bar{e}_1^I$ is obtained in an iterative process, i.e., $\bar{e}_1^I$ is the guessed output $(\hat{e}_1^J)$ (the optimal one of a previous iteration). In the first iteration, no guessed output is used and $R(j,i) = 0$ for $1 \leq i \leq I$ and $1 \leq j \leq J$.

The base of recursion for $0 \leq j \leq J$ and for all possible word e is:

$$\begin{aligned} T(e,1) &= p_b(e|\$) \\ Q(e,1,j) &= t(f_j|e_0) \cdot a(0|j,J,I) \\ &\quad + t(f_j|e) \cdot a(1|j,J,I) \end{aligned}$$

where $p_b(e|\$)$ is the language model probability when e is the first word in the sentence to be generated.

The array T accounts for the language model score and the array Q accounts for the IBM-Model 2 translation model score associated to $\hat{e}_1^I$. In each step of the trellis, the array Q in some way explain how the current generated word e_i affects all the words in the input sentence f_1^J. This is the heuristic point of this approach. It might be assumed that the "correct" (best hypothesis) according to the IBM-Model 2 translation model is the one that maximizes the

Viterbi alignment between the sentence pair $(f_1^J, \hat{e}_1^I)$ (input sentence, hypothesis sentence). The problem is that, in order to compute the best IBM-Model 2 Viterbi alignment between a pair of sentences (f_1^J, e_1^I), both completed sentences must be known. This is the reason why the IDPS algorithm computes the score in the heuristic way. The word $\hat{e}(e, i)$ is the best word in the previous step of the trellis to reach the word e at position i.

It is clear that, the decision process is not straightforward for the IBM-Model 2, it is even harder for the most complex IBM models.

The IDPS algorithm is outlined here:

INPUTS: t, a, p_b, f_1^J.

OUTPUT: $\text{argmax}_{e_1^I} \left(Pr_{LM}(e_1^I) Pr_{TM}(f_1^J | e_1^I) \right)$.

initialization

- Initialize array R to 0.
- Compute a first approximation $(\hat{e}_1^I)$ to be the solution of equation (7) by using the search procedure.

iteration

- **While not** convergence **do**
  1. Update array R according to the previous $\hat{e}_1^I$ using equation (9)
  2. Compute a new approach $(\hat{e}_1^I)$ to be the solution of equation (7) by using a prefix that is built during the current iteration.

  end of While
- **Return** $\hat{\mathbf{e}}_1^I$ with best score.

To clarify the search procedure, an example is depicted in Figures 1, 2 and 3. Figure 1 shows a common initial situation that could be found at step i of the trellis search. For example, in the states labeled with e', e'' and e''' at step $i-1$ (possible predecessor states to reach the state e at step i) the accumulated scores corresponding to arrays Q and T are stored. Figure 2 shows the computation that is carried out within the algorithm for the three previous different words to be considered. Let us suppose that the best score is obtained when the word e'' is used. Then the Figure 3 presents this case with the corresponding update of the score for word e at step i.

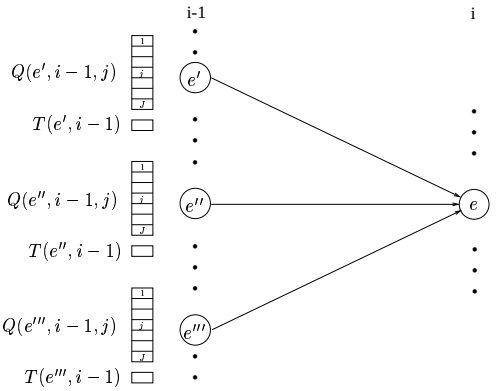

Figure 1: A common initial situation at step i in the trellis.

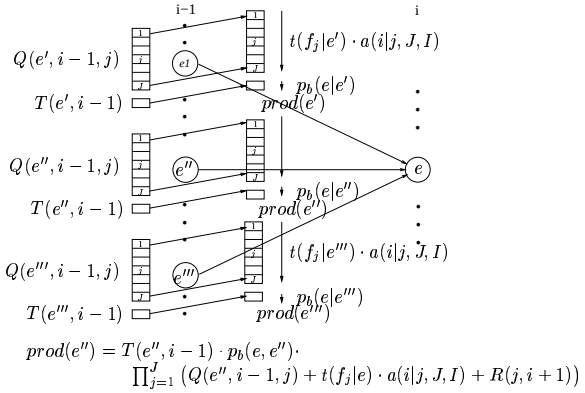

Figure 2: Search update for e in step i of the trellis using as previous possible words e', e'' and e'''.

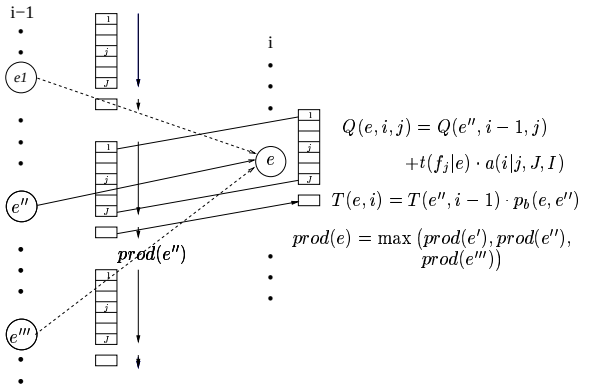

Figure 3: Final decision of values for word e in step i of the trellis and the update of corresponding values.

New features to the IDPS

As we pointed out in the previous section, the IDPS algorithm cannot directly provide the hypothesis with the best Viterbi alignment score in the decision making process. One way of dealing with this problem is that once the set of possible hypothesis is computed to choose the hypothesis that provides the best Viterbi alignment score instead of choosing the one that could be obtained with the decision process outlined in the previous section. At the end of the decision process we know the whole hypothesis sentence, and it is then perfectly possible to compute the Viterbi alignment score associated to the sentence pair $(f_1^J, \hat{e}_1^I)$ for all possible $\hat{e}_1^I$ obtained.

To include this approximation to the previous algorithm, we only need to do the following in the IDPS algorithm:

1. Build the trellis search following the formulation for the IDPS algorithm.
2. Once the set of possible hypothesis of length I is obtained, choose the hypothesis that provides the best

Viterbi alignment score according to the translation model used.

3. Update the array R according to the best Viterbi hypothesis and go to step 1 to continue the iterative process.

The new IDPS algorithm will be:

INPUTS: M, t, a, d, n, p_1, p_b, f_1^J.

OUTPUT: $\text{argmax}_{e_1^I} \left(Pr_{LM}(e_1^I) Pr_{TM}(f_1^J|e_1^I)\right)$.

initialization

- Initialize array R to 0.
- Compute a first approximation ($\hat{e}_1^I$) to be the solution of equation (7) by using the search procedure.

iteration

- **While not** convergence **do**
  1. Compute the Viterbi alignment score for model M of all possible hypotheses.
  2. Choose the best hypothesis $\hat{e}_1^I$ according to the best Viterbi alignment score provided by the Model M.
  3. Update the array R according to this Viterbi hypothesis.
  4. Compute a new approach ($\hat{e}_1^I$) to the solution by using a prefix that is built during the current iteration and a suffix corresponding to the previous Viterbi hypothesis.

  end of While
- **Return** $\hat{e}_1^I$ with best Viterbi alignment score.

The input of the algorithm will be determined by the model M we want to use. For example, for IBM-Model 3 we need the *lexicon model* $t(f_j|e_i)$, the *distortion model* $d(j|i,m,l)$, the *fertility model* $n(\phi_i|e_i)$ and the probability p_1 that the null word e_0 will be aligned to some input words ($p_0 = 1 - p_1$). For all cases the language model probability $p_b(e_{i-1}|e_i)$ is needed. In order to solve the equation (3) with the IBM-Model 3, the formula to apply is (Brown et al., 1993):

$$Pr(a, f_1^J|e_1^I) = \binom{m-\phi_0}{\phi_0} p_0^{m-2\phi_0} p_1^{\phi_0} \cdot \prod_{i=1}^{I} n(\phi_i|e_i) \prod_{i=0}^{I} \phi_i! \cdot \prod_{j=1}^{J} n(f_j|e_i) \cdot \frac{1}{\phi_0!} \cdot \prod_{j:a_j \neq 0}^{J} d(j|i,J,I)$$

The first advantage of this new approach is that it is feasible to deal with the rest of the models proposed in (Brown et al., 1993). For example, to deal with IBM-Model 3, we only need to compute the Viterbi alignment score for this model in step 1 of the previous algorithm instead of the IBM-Model 2 Viterbi alignment score. We could proceed in the same why for the IBM-Model 4 and IBM-Model 5[1].

As is stated in (Brown et al., 1993) and in (Al-Onaizan et al., 1999) it is not possible to compute as exactly the best Viterbi alignment for IBM-Model 3 to IBM-Model 5, as in models IBM-Model 1 and IBM-Model 2. In the training process, the Viterbi alignment for IBM-Model 3 to IBM-Model 5 (Och and Ney, 2000a) is computed by first computing the Viterbi alignment according to IBM-Model 2 and afterwards making slight changes in the alignment (called *moves* and *swaps*) in order to get a neighborhood of possible alignments. Once this neighborhood has been obtained, the Viterbi alignment is computed using the *Hill-Climbing* algorithm introduced in (Al-Onaizan et al., 1999). This algorithm is also used in the translation process when the Viterbi alignment for IBM-Model 3 to IBM-Model 5 is computed. This is why the search procedure (or construction of the search trellis) is still based on the parameters of the IBM-Model 2. On the other hand, as we stated in the previous section, it is not straightforward to compute search procedure on the basis of the parameters of the most complex IBM models.

Obviously, this new approach is also heuristic because the search principle of the IDPS is still maintained.

Pruning techniques

The computational time complexity of the IDPS algorithm (García-Varea et al., 1998) is: $O(J \cdot I_{max} \cdot n_I \cdot |V(\mathcal{E})|)$ where: J is the input length sentence, I_{max} is the maximum output sentence length, n_I is the number of iterations[2] and $|V(\mathcal{E})|$ is the size of the output vocabulary. Obviously, the most important factor in the asymptotic cost is the size of the output vocabulary.

To speed up the search, we have introduced two pruning techniques. These two techniques are:

Beam-search pruning

This is the well-known *beam search* which is a direct analogy to the data-driven search organization used in continuous speech recognition (Ney et al., 1992). A beam search value is established a priori. During the dynamic programming search, those partial scores that have a higher score than the best one plus the beam search value are discarded. This avoids those hypotheses that are far from being the best one at the end of the process.

Alignment-based pruning

The second pruning technique is an *alignment-based* technique. Without any pruning technique, every target word e in each step of the trellis search would be considered as a possible word in the final translated sentence. Considering the training parameters obtained in the EM training process of the IBM translation models as well as the results presented in (García-Varea et al., 1998), we have observed that the feasible words that will belong to the translated

[1] For more details about the mathematics of the IBM models see (Brown et al., 1993).

[2] In the experiments we carried out this value is fixed to 3. Higher values do not yield in better results.

sentence are those that have a high translation probability according to the translation model that is used. In other words, all the those words e that are connected/aligned to the words in the input sentence f_1^J at the end of the training process are highly probable candidates to be part of the translated sentence. Taking these considerations into account, we restrict the possible words $e \in V(\mathcal{E})$ to the ones that are aligned at least once to the words in $f_1^J = f_1 f_2 \cdots f_J$. In this new set of possible e words to be considered, we should include those words that actually did not generate any source word f during the training process. These words are called *0-fertility* words (Al-Onaizan et al., 1999). These 0-fertility words should be included in order to take into account those words that do not generate any f_j word but actually form part of the translated sentence. In the experiments that were carried out, the number of words to be considered in each step of the trellis was substantially reduced. The advantage of this pruning technique is that the set of "good" candidates e for an input sentence could be obtained automatically and a priori using the Viterbi alignment computed in the training process. In other words, this pruning technique does not overload the translation process and the asymptotic computational time is not increased. What is more, it is decreased by a factor of 100 as we will show in the experiments and results section.

Experiments and results

We selected the "Tourist Task" (Amengual et al., 1996) to experiment with the search algorithm proposed here. The general domain of the task was a visit by a tourist to a foreign country. The task used for the experiments reported here corresponded to a scenario of human-to-human communication situations at the reception desk of a hotel. This task provided a small "seed corpus" from which a large set of sentence pairs was generated in a semi-automatic way (Amengual et al., 1996). From the different pairs of languages that were generated, only Spanish to English was considered for this work. The parallel corpus consisted of 500,000 sentence pairs (171,481 different sentence pairs). The input and output vocabulary sizes were 689 and 514, and the average input and output sentence lengths were 9.7 and 9.9, both respectively.

A sub-corpus of 10,000 random sentence pairs was selected for training purposes from the above corpus. Testing was carried out with 1,000 input random sentences generated independently from the training set. The training and test sets were disjoint. The training of the different translation models was carried with the GIZA (Al-Onaizan et al., 1999) software and its extension GIZA++ (Och and Ney, 2000a; Och and Ney, 2000b) using 15 iterations of the maximization-expectation algorithm. In order to compute the *aligned-based pruning*, we used the 10,000 training IBM-Model 5 Viterbi alignments obtained by the GIZA++ software. The language model used in the experiments was a 3-gram model. This language model was inferred using the 10,000 English training sentences.

For the evaluation of the translation quality, we used the automatically computable Word Error Rate (WER) which corresponds to the edit distance between the produced translation and a predefined reference translation.

One shortcoming of the WER is the fact that it requires perfect word order. This is particularly a problem for the Tourist Task, where the word order of the Spanish-English sentence pair can be quite different. As a result, the word order of the automatically generated target sentence can be different from that of the target sentence but acceptable, and thus the WER measure alone could be misleading. In order to overcome this problem, as additional measure, we introduce the position-independent word error rate (PER). This measure compares the words in the two sentences *without* taking the word order into account. Words that have no matching counterparts are counted as substitution errors. Depending on whether the translated sentence is longer or shorter than the target translation, the remaining words result in either insertion or deletion errors in addition to substitution errors. The PER is guaranteed to be less than or equal to the WER.

The translation results in terms of error rates are shown in Table 1.

	Alig. Prunning		No Prunning	
Model	WER	PER	WER	PER
No Vit.	19.09	15.24	20.37	16.27
Vit. IBM-Model 2	20.66	15.84	21.68	16.72
Vit. IBM-Model 3	22.76	17.61	22.80	17.19
Vit. IBM-Model 4	14.81	12.80	15.75	13.54
Vit. IBM-Model 5	15.80	12.81	16.73	13.33

Table 1: Translation results for the 1000 test sentences with and without the alignment-based prunning technique. The first row shows the translation results obtained by using the original IDPS algorithm. The following rows show the translation results obtained with the new version of the IDPS for different IBM translation models.

According to Table 1 the translation results using the baseline IDPS and the Viterbi IBM-Model 2 are mainly what we expected. As in (Tillmann, 2001) the translation results using the Viterbi IBM-Model 4 are slightly better than those obtained with Viterbi IBM-Model 5. The best results were obtained using the Viterbi IBM-Model 4 score with an improvement of about 5 points of WER with respect to the baseline results. In Table 1, it can be observed that the alignment-based prunning technique not only substantially reduces the speed of the algorithm but even slightly increases the translation quality. That means that the pruning directs the search through the most probable paths in the trellis, discarding those possible hypotheses that could provide a better final score but that are actually worse in terms of translation quality.

In Table 2, it can be observed that there is a significant savings in the computational cost of the algorithm when the alignment-based pruning technique is applied. The table shows the average number of states in the trellis during the search procedure (i.e. the number of times that equation (8) should be computed). These values are computed by multiplying the input test length sentence average (which

is 11.68) by the the number of words to be considered in each state of the trellis. The number of possible candidate words on average per sentence to be considered during the search process are: 514 when no pruning is applied (this number is equal to the size of the English vocabulary) and 7.49 when the proposed technique is used. The reduction factor is about 100.

	# of states
No pruning	6008.62
Alignment-based	87.59

Table 2: Average number of states to be considered during the search process with and without pruning for the translation experiments shown in Table 1.

Conclusions

In this paper we have included new features to the IDPS algorithm. We have included the possibility of dealing with the most complex IBM models (from 3 to 5) while maintaining the decision making process based on the IBM-Model 2. An alignment-based pruning technique is also included in order to improve the speed up of the algorithms.

According to the results presented in the previous section, significantly better translation results are obtained when the Viterbi IBM-Model 4 scoring is applied. There is also a significant improvement in the speed of the algorithm when the proposed pruning technique is used.

For future work, we plan to investigate how to directly include the parameters of the most complex IBM models (IBM-Model 3 to IBM-Model 5) in the dynamic-programming based search. It would also be valuable to include more complex pruning techniques paying special attention to the loss of the translation quality.

Acknowledgements

This work has been partially supported by Spanish CICYT under grants TIC 1FD1997-1433 and TIC2000-1599-C02.

References

Al-Onaizan, Y., Curin, J., Jahr, M., Knight, K., Lafferty, J., Melamed, I. D., Och, F. J., Purdy, D., Smith, N. A., and Yarowsky, D. (1999). Statistical machine translation, final report, JHU workshop. `http://www.clsp.jhu.edu/ws99/projects/mt/final_report/mt-final-report.ps`.

Amengual, J., Benedí, J., Castao, M., Marzal, A., Prat, F., Vidal, E., Vilar, J., Delogu, C., di Carlo, A., Ney, H., and Vogel, S. (1996). Definition of a machine translation task and generation of corpora. Technical report d4, Instituto Tecnológico de Informática. ESPRIT, EuTrans IT-LTR-OS-20268.

Brown, P., Cocke, J., Pietra, V. D., Pietra, S. D., Jelinek, J., Lafferty, J., , Mercer, R., and Roossina, P. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

García-Varea, I., Casacuberta, F., and Ney, H. (1998). An iterative, dp-based search for statistical machine translation. In *Procs. of the 5th International Conference on Spoken Language Processing (ICSLP'98)*, pages 1235–1138, Sydney, Australia.

Ney, H., Mergel, D., Noll, A., and Paeseler, A. (1992). Data driven search organization for continuous speech recognition in the spicos system. *IEEE Trans. on Signal Processing*, 40(2):272–281.

Niessen, S., Vogel, S., Ney, H., and Tillmann, C. (1998). A DP-based search algorithm for statistical machine translation. In *COLING-ACL '98: 36th Annual Meeting of the Association for Computational Linguistics and 17th Int. Conf. on Computational Linguistics*, pages 960–967, Montreal, Canada.

Och, F. J. and Ney, H. (2000a). A comparison of alignment models for statistical machine translation. In *COLING '00: The 18th Int. Conf. on Computational Linguistics*, pages 1086–1090, Saarbrücken, Germany.

Och, F. J. and Ney, H. (2000b). Improved statistical alignment models. In *Proc. of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hongkong, China.

Tillmann, C. (2001). *Word Reordering and Dynamic Programming based Search Algorithm for Statistical Machine Translation*. PhD thesis, Lehrstuhl für Informatik VI, RWTH Aachen Germany.

Tillmann, C., Vogel, S., Ney, H., and Zubiaga, A. (1997a). A dp based search using monotone alignements in statistical translation. In *Procs. of the Association of Computational Linguistics*, pages 289–296, Madrid, Spain.

Tillmann, C., Vogel, S., Ney, H., Zubiaga, A., and Sawaf, H. (1997b). Accelerated dp based search for statistical translation. In *Procs. of the Fifth European Conf. on Speech Communication and Technology*, pages 2267–2670, Rhodos, Greece.

Vogel, S., Ney, H., and Tillmann, C. (1996). Hmm-based word alignment in statistical translation. In *Procs. of the Int. Conf. Computational Linguistics*, pages 836–841, Copenhagen, Denmark.

Wang, Y.-Y. and Waibel, A. (1997). Decoding algorithm in statistical translation. In *Proc. 35th Annual Conf. of the Association for Computational Linguistics*, pages 366–372, Madrid, Spain.