

ACCELERATED DP BASED SEARCH FOR STATISTICAL TRANSLATION

C. Tillmann, S. Vogel, H. Ney, A. Zubiaga, H. Sawaf
Lehrstuhl für Informatik VI, RWTH Aachen
D-52056 Aachen, Germany

ABSTRACT

In this paper, we describe a fast search algorithm for statistical translation based on dynamic programming (DP) and present experimental results. The approach is based on the assumption that the word alignment is monotone with respect to the word order in both languages. To reduce the search effort for this approach, we introduce two methods: an acceleration technique to efficiently compute the dynamic programming recursion equation and a beam search strategy as used in speech recognition. The experimental tests carried out on the Verbmobil corpus showed that the search space, measured by the number of translation hypotheses, is reduced by a factor of about 230 without affecting the translation performance.

1. INTRODUCTION

In this paper, we address the search problem in statistical translation and present fast algorithms for the presented task. To review the statistical translation approach, we consider a source (‘French’) string $f_1^J = f_1 \dots f_j \dots f_J$, which is to be translated into a target (‘English’) string $e_1^I = e_1 \dots e_i \dots e_I$. Among all possible target strings, we will choose the one with the highest probability which is given by the Bayes’ decision rule:

$$\begin{aligned} \hat{e}_1^I &= \arg \max_{e_1^I} \{Pr(e_1^I | f_1^J)\} \\ &= \arg \max_{e_1^I} \{Pr(e_1^I) \cdot Pr(f_1^J | e_1^I)\} . \end{aligned}$$

$Pr(e_1^I)$ is the language model of the target language, whereas $Pr(f_1^J | e_1^I)$ is the string translation model. The argmax operation denotes the search problem. In this paper, we address the search procedure, i.e. an algorithm to perform the argmax operation in an efficient way. The string translation probability $Pr(f_1^J | e_1^I)$ defines a correspondence between the words of the target sentence and the words of the source sentence. In this paper we assume a dependence where each source word corresponds to *exactly one* target word. Models describing these types of dependencies are referred to as *alignment models* [1, 2, 3, 6]. For the translation model we use a so-called HMM-based alignment model as described in [3, 6], where the probability $p(a_j | a_{j-1})$ of aligning the j -th word in the source sentence to the a_j in the target sentence depends on the previous alignment a_{j-1} for position $j-1$. The problem formulation is similar to that of

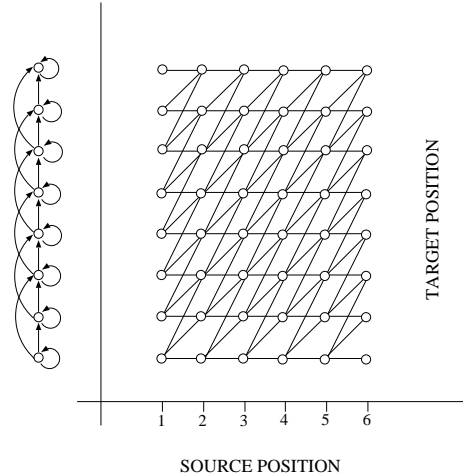


Figure 1: Illustration of alignments for the monotone HMM.

the time alignment problem in speech recognition, where the so-called Hidden Markov models have been successfully used for a long time. To train the parameters we perform a maximum likelihood estimation of the translation parameters using the EM algorithm[6]. Details are omitted here due to space limitations.

2. SEARCH ALGORITHM FOR TRANSLATION

2.1. Baseline DP Algorithm

The system uses two knowledge sources, namely a bigram language model $p(e|e')$, which is given in terms of the conditional probability of observing word e_i given the predecessor word e_{i-1} , and a HMM-based translation model. We can use dynamic programming to find the target sentence e_1^I with the highest a-posteriori probability in Bayes’ decision rule. The search procedure presented here is based on the assumption that the word alignment is monotone with respect to the word order in both languages [5]. The search problem can be understood as finding a path through a network with a uniform trellis structure. When searching the best path through the trellis, we find an alignment for each word position in the source sentence j with a position in the target sentence a_j and a word e_{a_j} at this position. Due to the monotony of our alignment model and the bigram language model, we have only first-order type dependencies such that the local probabilities (or costs when using the negative log-

arithmetic of the probabilities) depend *only* on the arcs (or transitions) in the lattice. The local probabilities or scores for the arcs are computed using the language and translation model. Each possible index triple (i, j, e) defines a grid point in the lattice. Using this formulation of the search task, we can now use the method of dynamic programming (DP) to find the best path through the lattice, where we use the auxiliary quantity:

$Q(i, j, e)$: probability of the best partial path which ends in the grid point (i, j, e) .

For the used monotone translation model the auxiliary quantity satisfies the following DP recursion equation:

$$Q(i, j, e) = p(f_j|e) \cdot \max_{\delta} \{p(i|i-\delta) \cdot \max_{e'} p_{\delta}(e|e') \cdot Q(i-\delta, j-1, e')\},$$

where $p(f|e)$ is the translation probability for the French word f given the English word e and $p(i|i-\delta)$ is the alignment probability. The monotony assumption restricts the alignment probabilities $p(a_j|a_{j-1})$ to the three cases $\delta = 0, 1, 2$:

- $\delta = 0$: This case corresponds to a word repetition (i.e. a target word with more than one aligned source word). We have: $p_{\delta=0}(e|e') = 1$ iff $e = e'$.
- $\delta = 1$: This case is the regular one. We use the probability of the bigram language model: $p_{\delta=1}(e|e') = p(e|e')$.
- $\delta = 2$: (skip transition): This case corresponds to skipping a word, i.e. there is a word \tilde{e} in the target string with no aligned word in the source string. We have to carry out the following optimization over the non-aligned word \tilde{e} :

$$p_{\delta=2}(e|e') = \max_{\tilde{e}} [p(e|\tilde{e}) \cdot p(\tilde{e}|e')].$$

This maximization is done beforehand and the result is stored in a table.

The DP equation is evaluated recursively to find the best partial path to each gridpoint (i, j, e) . The complexity of the algorithm is $J \cdot I_{max} \cdot E^2$, where E is the size of the target language vocabulary and I_{max} is the maximum length of the target sentence considered. We present two acceleration techniques for the efficient computation of the DP equation: accelerated search and beam search.

2.2. Accelerated Search

To find the best partial path to each gridpoint (i, j, e) , the arcs in the lattice leading to (i, j, e) are processed in a specific order so that the computation can be stopped whenever it is sure that no better partial path to (i, j, e) exists. By carrying out these computations at each gridpoint, it is guaranteed to find the best path through the lattice. Although the complexity of the algorithm is not guaranteed to be reduced in the worst case, we obtain significant computational savings in the average case as will be shown in the experiments.

The details of the algorithm are as follows. For each successor word e , we store the best M predecessor words

$e_1^{(\delta)} \dots e_M^{(\delta)}$ in a list of candidates ranked by $p_{\delta}(e|e_i^{(\delta)})$, e.g. $M = 10$. The computation is carried out similarly for each of the cases $\delta = 1, 2$. For $\delta = 0$, no optimization is needed because in this case there is only one predecessor gridpoint. We expect the best score $Q(i, j, e)$ for the gridpoint (i, j, e) to result from the M predecessor words and perform the optimization over this restricted set beforehand. The best score that results from the M predecessor words is denoted by $\hat{Q}(i, j, e)$. To guarantee the optimality of the score $\hat{Q}(i, j, e)$, we sort the arcs leading to gridpoint (i, j, e) according to the scores of the predecessor gridpoints $(i-\delta, j-1, e')$. The probability of an arc between the two gridpoints is:

$$\mathcal{C}_{(\delta, i, j)}(e, e') = p_{\delta}(e|e') \cdot p(f_j|e) \cdot p(i|i-\delta)$$

For the words e' that are not in the candidate list, we define an upper bound $t_{\delta, e}$ for $p_{\delta}(e|e')$:

$$t_{\delta, e} = \min_{e_1^{(\delta)} \dots e_M^{(\delta)}} p_{\delta}(e|e_i^{(\delta)}).$$

Further we define:

$$\bar{\mathcal{C}}_{(\delta, i, j)}(e) = t_{\delta, e} \cdot p(f_j|e) \cdot p(i|i-\delta).$$

$\bar{\mathcal{C}}_{(\delta, i, j)}(e)$ is an upper bound of $\mathcal{C}_{(\delta, i, j)}(e, e')$ for all e' not in the candidate list:

$$\mathcal{C}_{(\delta, i, j)}(e, e') \leq \bar{\mathcal{C}}_{(\delta, i, j)}(e).$$

When processing the arcs leading to (i, j, e) in the order of the scores $Q(i-\delta, j-1, e')$, there are two computational steps. When a word e' is encountered for which $\mathcal{C}_{(\delta, i, j)}(e, e') \cdot Q(i-\delta, j-1, e') > \hat{Q}(i, j, e)$ holds then $\hat{Q}(i, j, e)$ is updated:

$$\hat{Q}(i, j, e) := \mathcal{C}_{(\delta, i, j)}(e, e') \cdot Q(i-\delta, j-1, e').$$

In the second step the following condition is tested:

$$\bar{\mathcal{C}}_{(\delta, i, j)}(e) \cdot Q(i-\delta, j-1, e') < \hat{Q}(i, j, e).$$

If the condition holds the computation is terminated with $Q(i, j, e) = \hat{Q}(i, j, e)$. Thus the average complexity for finding the optimal score at each gridpoint (i, j, e) is reduced from $J \cdot I_{max} \cdot E^2$ to $J \cdot I_{max} \cdot E \cdot (\log E + M + \bar{E})$, where \bar{E} is the average number of processed arcs in the sorted lists and $\log E$ is due to the sorting operation of the arc probabilities.

2.3. Beam Search

The dynamic programming approach provides an efficient technique for performing the above search problem with the help of a pruning strategy. There is a direct analogy to the data-driven search organization used in continuous speech recognition [4]. The DP based translation algorithm proceeds from left to right along the positions j in the source sentence. The construction of the hypotheses is guided by the language model, the translation model and the alignment model. During the translation process the search space is dynamically constructed rather than using a static predefined search space. The computational cost of the algorithm is linearly proportional to the number of calculated hypotheses and independent of

the size of the overall search space. The size of the local search space at position j depends on the ambiguity of the translation task at that position.

For each position j we maintain a list of hypotheses (i, e) for different positions i and target words e . This set of hypotheses is the so-called beam. Denoting the best scored hypothesis for the word position j by:

$$Q_{Beam}(j) := \max_{(i,e)} Q(i, j, e),$$

we retain only hypotheses with a score close to the best hypotheses for the further consideration. Thus we prune a hypothesis (i, j, e) if:

$$Q(i, j, e) < f_{Beam} \cdot Q_{Beam}(j),$$

where the number of surviving hypotheses is controlled by the threshold f_{Beam} .

3. EXPERIMENTAL RESULTS

3.1. Database

The search algorithms proposed in this paper were tested on the ‘‘Verbmobil Task’’ [7]. The Verbmobil task is an appointment scheduling task, where two subjects are each given a calendar and asked to schedule a meeting. The translations are carried out from German to English. The original corpus consisted of 16 857 German-English sentence pairs. We removed all sentence pairs containing words, that occurred only once in the training corpus (so-called singletons) and obtained a training set of 14210 sentences pairs. This set was splitted into 13737 sentences for training and 473 sentences for testing. From this test set, 92 test sentences were filtered out for which the monotony constraint was more or less satisfied or could easily be obtained by simple word reorderings. The statistics of the corpus are given in Table 1.

For the word reordering we used a simple parser that reordered the words in the source sentences to get the German word order closer to the typical English word order. This was done mainly for the words belonging to the verb phrase and the German negation word ‘‘nicht’’. For the reordering, which is not perfect yet, all sentences had been tagged with parts-of-speech information. The reordering was performed for both the training and the test sentences.

The lexicon models were trained with the reordered source sentences. To improve the lexicon probabilities $p_T(f|e)$ (T=trained), we used a bilingual German-

Table 1: Training and test conditions for the Verbmobil task.

	German	English
Vocabulary	1,889	1,358
Training: Sentences	13 737	
Words	140,702	139,903
Test: Sentences	92	
Words	652	665

Table 2: Examples of the Verbmobil task: O= original sentence after reordering, R= reference translation, A= automatic translation.

O:	Wo wollen wir treffen uns denn ?
R:	Where shall we meet ?
A:	Where shall we meet then ?
O:	Ich sagte schon , ich habe da eine Besprechung.
R:	As I said , I have got a meeting then.
A:	I said before , I have got a meeting.
O:	Das wu"rd mir ganz gut passen.
R:	That would be fine with me.
A:	That would suit me fine.
O:	Und wann sollen wir das Abendessen danach machen ?
R:	And when shall we have the supper afterwards ?
A:	And when shall we could have dinner afterwards ?
O:	Wir mu"sten vereinbaren da mehrere Termine jetzt f"ur November und Dezember dreiundneunzig .
R:	We have to arrange several appointments for November and December nineteen-ninety-three .
A:	We should fix that I have several dates now for November and December nineteen-ninety-three.

English dictionary. For each word e in the English vocabulary, we created a list of German translations f according to this dictionary. The lexicon probability $p_D(f|e)$ for the dictionary entry (e, f) is defined as:

$$p_D(f|e) = \begin{cases} 0 & \text{if } (e, f) \text{ not in lexicon} \\ \frac{1}{N_e} & \text{if } (e, f) \text{ in lexicon} \end{cases},$$

where N_e is the number of German words listed as translations of the English word e . The created lexicon was linearly combined with the lexicon probabilities $p_T(f|e)$ (using the interpolation parameter λ) to obtain the smoothed probabilities $p(f|e)$:

$$p(f|e) = (1 - \lambda) \cdot p_D(f|e) + \lambda \cdot p_T(f|e).$$

For the translation experiments, we used the value $\lambda = 0.5$.

3.2. Experimental Tests

We used the Levenshtein distance between the automatic translation and the reference translation as a measure of the translation errors. Word Error Rates (WER) are reported at the word level along with the number of insertions (INS) and deletions (DEL). A weak point of the WER is the fact that word ordering is not taken into account appropriately. In order to overcome this problem, we introduce as a new measure the position-independent word error rate (PER) that is computed at the word level,

Table 3: Effect of the number M of preselected candidates e' in the accelerated DP algorithm on the computation time (Verbmobil task: 92 sentences = 652 words).

M	CPU Time[sec]	WER[%]
1	652	25.9
3	500	25.9
5	480	25.9
7	539	25.9
10	547	25.9
all: $M = 1358$	21132	25.9

Table 4: Effect of the beam threshold on the search effort (active overall hypotheses, active words, active positions) and word error rates (WER,PER) (Verbmobil task: 92 sentences = 652 words).

beam threshold	CPU time [sec]	average number of active			position-independent		position-dependent	
		Hypotheses	Words	Positions	INS/DEL [%]	PER[%]	INS/DEL [%]	WER[%]
0.0	27	1.0	1.0	1.0	4.5/8.5	28.3	11.1/6.9	32.8
1.0	31	1.5	1.2	1.3	3.8/7.7	24.4	10.8/6.8	29.8
2.5	46	3.1	2.0	2.1	4.9/5.4	23.5	7.7/7.2	26.8
5.0	91	11.1	6.2	3.5	6.6/5.0	23.2	7.1/8.6	25.9
7.5	211	40.2	20.8	4.7	6.6/4.5	23.2	6.6/8.7	25.9
10.0	530	126.8	57.7	5.7	6.6/4.5	23.2	6.6/8.7	25.9

too. We do not take into account the word order but count only the number of times identical words occur in both sentences. Words that do not match are counted as substitutions. Depending on whether the translated sentence is longer or shorter than the reference translation, the rest of the words are either insertions or deletions. The PER is guaranteed to be less than or equal to the WER. Admittedly, these two error criteria are not perfect measures. But they can be automatically computed and are easy to use.

Both accelerating techniques were tested using the test settings as described above. The language model perplexity of the bigram language model used was 17.3. For the 92 test sentences, we obtained a WER of 25.9%. This WER was obtained using the accelerated search that guarantees to find the best path through the translation lattice.

Table 2 shows translation examples. The German sentences are given after reordering. For many sentences we obtain acceptable translations although the word error rate is high. The effect of the number of candidates M on the computation time (SGI workstation Indy R4000) is shown in Table 3. The minimum CPU time is obtained for $M = 5$.

Table 4 shows the results of the translation experiments for the beam search DP approach. The search space is given as a function of the beam threshold f_{beam} , where the negative logarithm of the value f_{beam} is reported. The search space is given in terms of the average number of active overall hypotheses, active words and active positions. The WER varies between 32.8% and 25.9%. The reason is that the beam search approach can produce search errors that result in additional translation errors.

4. CONCLUSION

In this paper we presented fast search algorithms based on dynamic programming for statistical translation. The approach is based on the assumption that the word alignment is monotone with respect to the word order in both languages. The presented beam search technique resulted in a reduction of the search space by a factor of about 230 without loss in translation performance. For the Verbmobil task, a sentence is translated within a few seconds. Future plans involve the combination of accelerated search and beam search as well as the incorporation

of a trigram language model.

Acknowledgement. This work has been partially supported by the German Federal Ministry of Education, Science, Research and Technology under the contract number 01 IV 701 T4 (Verbmobil).

REFERENCES

1. A. L. Berger, P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, J. R. Gillett, J. D. Lafferty, R. L. Mercer, H. Printz, and L. Ures, "The Candide System for Machine Translation", in *Proc. ARPA Human Language Technology Workshop*, Plainsboro, NJ, Morgan Kaufmann Publishers, San Mateo, CA, pp. 152-157, March 1994.
2. P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer, "The Mathematics of Statistical Machine Translation: Parameter Estimation", in *Computational Linguistics*, Vol. 19, No. 2, pp. 263-311, 1993.
3. I. Dagan, K. W. Church, and W. A. Gale, "Robust Bilingual Word Alignment for Machine Aided Translation", in *Proc. of the Workshop on Very Large Corpora*, Columbus, OH, pp. 1-8, 1993.
4. H. Ney, D. Mergel, A. Noll, A. Paeseler, "Data Driven Search Organization for Continuous Speech Recognition". *IEEE Trans. on Signal Processing*, Vol. SP-40, No. 2, pp. 272-281, February 1992.
5. C. Tillmann, S. Vogel, H. Ney and A. Zubiaga, "A DP based Search Using Monotone Alignments in Statistical Translation", in *Proc. of the 35th Annual Conf. of the Association for Computational Linguistics*, Madrid, Spain, July 1997.
6. S. Vogel, H. Ney, and C. Tillmann, "HMM-Based Word Alignment in Statistical Translation", in *Proc. of the International Conference on Computational Linguistics*, pp. 836-841, Copenhagen, Denmark, August 1996.
7. Wolfgang Wahlster, "Verbmobil: Translation of Face-to-Face Dialogs", in *Proc. of the MT Summit IV*, pp. 127-135, Kobe, Japan, 1993.