Improvements in Dynamic Programming Beam Search for Phrase-based Statistical Machine Translation

Richard Zens[*] *and Hermann Ney*

Human Language Technology and Pattern Recognition
Lehrstuhl für Informatik 6, Computer Science Department
RWTH Aachen University, D-52056 Aachen, Germany
{zens,ney}@cs.rwth-aachen.de

Abstract

Search is a central component of any statistical machine translation system. We describe the search for phrase-based SMT in detail and show its importance for achieving good translation quality. We introduce an explicit distinction between reordering and lexical hypotheses and organize the pruning accordingly. We show that for the large Chinese-English NIST task already a small number of lexical alternatives is sufficient, whereas a large number of reordering hypotheses is required to achieve good translation quality. The resulting system compares favorably with the current state-of-the-art, in particular we perform a comparison with cube pruning as well as with Moses.

1. Introduction

We address the search problem for phrase-based statistical machine translation [1, 2, 3]. Search is the task of finding the target language sentence that maximizes the posterior probability given the source sentence. It is computationally expensive and requires an efficient and fine-tuned algorithm. Here, we will describe such an algorithm in more detail than typically found in the literature. We will describe the problem in depth and show the exact dynamic programming (DP) recursion, details of pruning and rest score estimation as well as the actual algorithm. The resulting system compares favorably well with the current state-of-the-art. We will also show that it is important to focus on alternative reorderings, whereas on the other hand already a small number of lexical hypotheses is sufficient. To investigate this, we will explicitly distinguish reordering and lexical hypotheses and analyze the search space. We also perform a comparison with cube pruning and we achieve the same performance with a simpler implementation.

[*]Richard Zens is now affiliated with Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043; zens@google.com.

2. Problem definition

Search is the task of finding the target language sentence e_1^I that maximizes the posterior probability given the source sentence f_1^J:

$$\hat{e}_1^{\hat{I}} \quad = \quad \operatorname*{argmax}_{I,e_1^I} \left\{ Pr(e_1^I | f_1^J) \right\}$$

Here, we are considering a phrase-based approach with a segmentation s_1^K into K phrases (which includes reordering). Assuming a log-linear model with components $h_m(\cdot)$ and scaling factors λ_m, we obtain:

$$\hat{e}_1^{\hat{I}} \quad = \quad \operatorname*{argmax}_{I,e_1^I} \left\{ \max_{K,s_1^K} \sum_{m=1}^{M} \lambda_m h_m(e_1^I, s_1^K; f_1^J) \right\}$$

We have to carry out a maximization over all possible target sentences e_1^I and over all possible segmentations s_1^K. As enumerating all target sentences is infeasible, we are facing a hard optimization problem. Nevertheless, we can exploit the structure of the models. We can interpret the search as a sequence of decisions $(\tilde{e}_k, b_k, j_k)$ for $k = 1, \ldots, K$. At each step we decide on a source phrase $\tilde{f}_k$ identified by its start and end positions $s_k = (b_k, j_k)$ and the corresponding translation $\tilde{e}_k$. To ensure that there are no gaps and no overlap, we keep track of the set of source positions that are already translated ('covered'). We will call this the *coverage* set $C \subseteq \{1, \ldots, J\}$. We can represent the search space as a graph where the arcs are labeled with the decisions $(\tilde{e}, j, j')$ and the states are labeled with the coverage sets C. The initial state is labeled with the empty coverage set. The goal state is labeled with the full coverage $C = \{1, \ldots, J\}$. Each path through this graph represents a possible translation of the source sentence, which is obtained by concatenating the target phrases $\tilde{e}$ along the path. Note that there are multiple paths representing the same translation with different phrase segmentations.

Some models do not have dependencies that cross phrase boundaries and can be computed for each phrase pair without context. In other words, these model scores depend only on a single arc in the graph. We will use $q_{\text{TM}}(\tilde{e}|j,j')$ to denote the weighted sum of all phrase model scores of an arc $(\tilde{e},j,j')$. This score consists of phrase-based models, word-based models, word and phrase penalty. Other models, however, take the context outside the phrase pair into account. Thus, their scores do depend on the decisions taken so far. Although in principal, these models could depend on the whole decision sequence, in practice, the models depend only on a small subset of the information. The n-gram language model (LM), for example, depends on the last $(n-1)$ words of the target sentence and the distortion model (DM) depends on the end position of the previous source phrase. Therefore, we distinguish the states according to the LM history and the end position of the last phrase. The score of this LM expansion weighted with the LM scaling factor is denoted as $q_{\text{LM}}(\tilde{e}|\tilde{e}')$. We use $q_{\text{DM}}(j'|j)$ to denote the weighted score of a jump from source position j to source position j'.

The states in the search space can be identified by a triple $(C,\tilde{e},j)$, where C denotes the coverage set, $\tilde{e}$ denotes the LM history, and j denotes the end position of the last source phrase. We will use the following terms:

- **Coverage hypothesis** C. A coverage hypothesis is identified by the coverage C.
- **Lexical hypothesis** $(\tilde{e},j)$. A lexical hypothesis is identified by the LM history $\tilde{e}$ and the source sentence position j.

The number of coverage hypotheses indicates how many alternative reorderings are investigated during the search. The number of lexical hypotheses per coverage hypothesis indicates the lexical alternatives that are taken into account. The score of an expansion of a state $(C,\tilde{e},j)$ with a phrase pair $(\tilde{e}',j'',j')$ is computed as

$$q_{\text{TM}}(\tilde{e}'|j'',j') + q_{\text{LM}}(\tilde{e}'|\tilde{e}) + q_{\text{DM}}(j''|j) \qquad (1)$$

The successor state is $(C\cup\{j'',\ldots,j'\},\tilde{e}\oplus\tilde{e}',j')$. Here, $\tilde{e}'\oplus\tilde{e}$ denotes the LM history after expanding the given history $\tilde{e}'$ with the phrase $\tilde{e}$. Also, we have to ensure that there is no overlap, i. e. that $C\cap\{j'',\ldots,j'\}=\emptyset$.

The search problem can be reformulated as finding the optimum path through this search graph. The size of the search graph is exponential in the source sentence length. It has been shown in [4] that the search problem is NP-hard. Here, we use two techniques to address this problem: dynamic programming [5] and beam search [6]. Using DP, we can reduce the number of paths that we have to explore in the search graph without giving up optimality. The idea of beam search is that at each step, we expand only the promising hypotheses and discard hypotheses that are unlikely to lead to the optimum solution. In contrast to DP, beam search may result in suboptimal solutions.

3. Dynamic programming

The DP solution described in this section is based on the algorithm for single-word based models of [7]. Here, we use a phrase-based version of this approach. The idea is that the search proceeds synchronously with the number of the already translated source positions. The resulting algorithm is similar to [8].

During the search, the translation is generated phrase by phrase, i. e. the search is monotonic in the target language. To permit reordering, the processing on the source side may be non-monotonic. Thus, we can jump forth and back within the source sentence. We avoid repeated computations by traversing the search graph in a topological order. Thus, before we process a node, i. e. expand the hypothesis, we have to make sure that we have visited all predecessors. We call the number of covered source positions of a hypothesis its cardinality c. We can easily guarantee the topological order by processing the nodes according to their cardinality.

As mentioned, the states of the search graph can be identified by a triple $(C,\tilde{e},j)$ with the coverage set C, the LM history $\tilde{e}$ and the end position j. We use the auxiliary quantity $Q(C,\tilde{e},j)$ to denote the maximum score of a path leading from the initial state to the state $(C,\tilde{e},j)$. The DP recursion equations are shown in Fig. 1. The computational complexity of the algorithm is in $\mathcal{O}\left(\sum_{c=1}^{J} L_s \cdot \begin{pmatrix} J \\ c \end{pmatrix} \cdot J \cdot V_e^{n-1} \cdot E\right)$ which can be simplified to $\mathcal{O}(J \cdot L_s \cdot E \cdot V_e^{n-1} \cdot 2^J)$ by taking into account that $\sum_{c=0}^{J} \binom{J}{c} = 2^J$. Here, V_e is the vocabulary size of the target language and n is the order of the LM, E denotes the maximum number of target phrases per source phrase.

4. Beam search

4.1. Pruning

As mentioned before, the complexity of the DP solution is exponential. To speed up the translation process, we use a beam search strategy [6] and apply pruning. We use two variants: threshold pruning and histogram pruning [9]. Histogram pruning means that we keep the best N hypotheses. Threshold pruning means that we keep a hypothesis only if its score is close to the best one. We use the following pruning strategies:

$$
\begin{aligned}
Q(\emptyset, \$, 0) &= 0 \\
Q(C, \tilde{e}, j) &= \max_{\substack{j'', j' : \{j', ..., j\} \subseteq C \\ j' \leq j < j' + L_s \\ \tilde{e}', \tilde{e}'' : \tilde{e}' \oplus \tilde{e}'' = \tilde{e}}} \Big\{ Q(C \setminus \{j', ..., j\}, \tilde{e}', j'') + q_{\text{TM}}(\tilde{e}''|j', j) + q_{\text{LM}}(\tilde{e}''|\tilde{e}') + q_{\text{DM}}(j'', j') \Big\} \\
\hat{Q} &= \max_{\tilde{e}, j} \Big\{ Q(\{1, ..., J\}, \tilde{e}, j) + q_{\text{LM}}(\$|\tilde{e}) + q_{\text{DM}}(j, J+1) \Big\}
\end{aligned}
$$

Figure 1: Dynamic programming recursion equations for non-monotone search.

Observation pruning. Here, we limit the number of translation options per source phrase. This is done before the actual search starts. Let τ_o denote the observation pruning threshold and let $q(j, j')$ denote the maximum score of any phrase translation $\tilde{e}$ of the source phrase $f_j, \ldots, f_{j'}$:

$$q(j, j') = \max_{\tilde{e}} \{ q_{\text{TM}}(\tilde{e}|j, j') + q_{\text{LM}}(\tilde{e}) \} \quad (2)$$

Here, $q_{\text{LM}}(\tilde{e})$ denotes the weighted LM score of target phrase $\tilde{e}$ without any given history, i. e. we use the unigram score of the first word, the bigram score of the second word and so on. We keep a target phrase $\tilde{e}$ as possible phrase translation of the source phrase $f_j, \ldots, f_{j'}$ if:

$$q_{\text{TM}}(\tilde{e}|j, j') + q_{\text{LM}}(\tilde{e}) + \tau_o \geq q(j, j') \quad (3)$$

Additionally, we apply observation histogram pruning with parameter N_o. Thus, if there are more than N_o target phrases for a particular source phrase, then we keep only the top N_o candidates.

Lexical pruning per coverage. Here, we consider all lexical hypotheses that have the same coverage C. The hypotheses may differ, for instance, in their LM history $\tilde{e}$ or the end position of the last phrase j. Here, we include an estimate for completing the hypothesis $R(C, j)$. A detailed description of the rest score estimate will be given in Sec. 4.2. Let τ_L denote the pruning threshold and let $Q(C)$ denote the maximum score of any hypothesis with coverage C:

$$Q(C) = \max_{\tilde{e}, j} \{ Q(C, \tilde{e}, j) + R(C, j) \} \quad (4)$$

Then, we keep a hypothesis with score $Q(C, \tilde{e}, j)$ if:

$$Q(C, \tilde{e}, j) + R(C, j) + \tau_L \geq Q(C) \quad (5)$$

Additionally, we apply histogram pruning with parameter N_L. Thus, if there are more than N_L hypotheses for a particular coverage C, then we keep only the top N_L candidates.

Coverage pruning per cardinality. Here, we consider all coverage hypotheses with a given cardinality c. As defined in Eq. 4, $Q(C)$ is the maximum score of any hypothesis with coverage C. We will use this value as score of the coverage hypothesis C. Let τ_c denote the pruning threshold, then we keep a coverage hypothesis with score $Q(C)$ if:

$$Q(C) + \tau_C \geq \max_{\substack{C:|C|=c, \\ \tilde{e}, j}} \{ Q(C, \tilde{e}, j) + R(C, j) \}$$

Additionally, we apply histogram pruning with parameter N_C. Thus, if there are more than N_C coverage hypotheses for a particular cardinality c, we keep only the top N_C candidates. Note that if we prune a coverage hypothesis C, we remove all lexical hypotheses with coverage C.

4.2. Rest score estimation

During pruning, we compare hypotheses which cover different parts of the source sentence. Here, it is important to use a rest score estimate for completing the hypothesis. Without such a rest score estimate, the search would first focus on the easy-to-translate part of the source sentence. This is of course undesirable. We use the rest score estimation described in [10, 11]. Note that this rest score estimation is related to the heuristic functions used in A* search algorithms. It is a requirement for the optimality of A* that the heuristic function is optimistic (or *admissible*). Here, we do not have this requirement and in fact for the LM we use a non-admissible rest score estimate. Nevertheless, the experiments show that including the LM score estimate is helpful.

We implemented two variants to estimate the rest score of a coverage $C \subset \{1 \ldots J\}$. The first variant follows [12] and estimates the rest score for *sequences* of uncovered source positions. The second variant follows [10] and estimates the rest score for individual uncovered source *positions*. We define $q^*(j, j')$ as the maximum score for translating source positions $j \ldots j'$:

$$
\begin{aligned}
q^*(j, j') &= \max \{ q(j, j'), \quad (6) \\
&\qquad \max_{j \leq k < j'} \{ q^*(j, k) + q^*(k+1, j') \} \}
\end{aligned}
$$

This recursive definition takes alternative phrase segmentations into account. The values of $q^*(j, j')$ can computed using DP in a straightforward way. To use this auxiliary function for the rest score estimation of a hypothesis with coverage $C \subset \{1 \ldots J\}$, we sum up the rest score estimates for the sequences of uncovered source positions:

$$R_{\text{Seq}}(C) = \sum_{(j,j') \in \bar{C}} q^*(j,j') \tag{7}$$

Here, $\bar{C}$ denotes the set of sequences of uncovered source positions. A sequence is only contained in $\bar{C}$ if it is of maximum length, formally:

$$\begin{aligned} \bar{C} &= \big\{(j,j') | j \leq j' \wedge \{j \ldots j'\} \subseteq \{1 \ldots J\} \setminus C \\ &\quad \wedge (j = 1 \vee j-1 \in C) \wedge (j' = J \vee j'+1 \in C)\big\} \end{aligned}$$

For the second variant, we define $q^*(j)$ as the maximum score for translating the source position j:

$$q^*(j) = \max_{j' \leq j \leq j''} \frac{q(j',j'')}{j''-j'+1} \tag{8}$$

To estimate the rest score of a hypothesis, we sum this quantity over the uncovered source positions:

$$R_{\text{Pos}}(C) = \sum_{j \in \{1 \ldots J\} \setminus C} q^*(j) \tag{9}$$

The computation of both variants is linear in the sentence length. In addition, we use a rest score estimation for the distortion model as described in [10]. Thus, we compute the minimum number of jumps to complete the hypotheses. The idea is that we jump from the end position of the current phrase j back to the first uncovered position. Then, we process the remaining part of the sentence from left to right and jump over all covered sequences. The rest score estimate for the distortion penalty model is:

$$\begin{aligned} j_0(C) &= \min\{j \mid 1 \leq j \leq J \wedge j \notin C\} \\ R_{\text{Dist}}(C,j) &= |j - j_0(C) + 1| + |C| - j_0(C) + 1 \end{aligned}$$

Here, $j_0(C)$ denotes the first uncovered position in the coverage set C. The jump distance from the current position j to the first uncovered position $j_0(C)$ is $|j - j_0(C) + 1|$. The number of covered positions to the right of position $j_0(C)$ is $|C| - j_0(C) + 1$. This is the number of source positions that we have to jump over to reach the sentence end. An algorithm for computing this value for a coverage represented as a bit vector is described in [10]. The effect of including the distortion rest score is equivalent to the method in [13].

The overall rest score estimate $R(C,j)$ is obtained as the sum of the distortion and TM/LM rest score estimate:

$$R(C,j) = \lambda_{\text{Dist}} \cdot R_{\text{Dist}}(C,j) + R_{\text{Seq}}(C) \tag{10}$$

5. Algorithm

We present the search algorithm including pruning in Fig. 3. The notation and functions used in this algorithm are shown in Fig. 2. We use CONTINUE and BREAK with the usual C/C++ semantics, i. e. CONTINUE will stop the current loop iteration and continue with the next iteration; BREAK will stop the whole loop.

The function 'pruneCardinality c' applies coverage and cardinality pruning to all hypotheses with cardinality c. In the function 'purgeCardinality c', we free the memory (except trace back information) of all hypotheses with cardinality c. For example, the coverage sets and the LM histories are not needed anymore and the memory can be reused.

Let $E(j',j)$ denote the set of phrase translations of the source phrase $\tilde{f} = f_{j'}, ..., f_j$. To avoid repeated computations, we generate the set of possible translations $E(j,j')$ for each phrase in the source sentence before the search along with their phrase model scores $q_{\text{TM}}(\tilde{e}|j,j')$. The auxiliary quantity $Q(C,\tilde{e},j)$ is the maximum score of a partial translation with coverage C, LM history $\tilde{e}$ and end position of the last source phrase j. In addition, we store backpointers $B(\cdot)$ to the previous best decision as well as the maximizing arguments $A(\cdot)$, i. e. the best target phrases. These are used to trace back the best path when the search is finished. For each cardinality c, we have a loop over all possible source phrase lengths l. Here, L_s denotes the maximum source phrase length. Then, we have a loop over the possible predecessor coverages C' with cardinality $c-l$. The next loop goes over all possible start position j, thus effectively we select a source phrase $\tilde{f} = f_j, ..., f_{j+l}$. We also check the 'no overlap' constraint in line 5. We generate the new coverage C and loop over all existing predecessor states $\tilde{e}', j'$ in line 7 and over all translation options $\tilde{e}''$ in line 10. Eventually, we compute the score of the expansion in line 12. If this is better than the existing one, we update this as well as the backpointer and the maximizing argument.

Pruning is applied after we generated all hypotheses of the current cardinality c in line 19. Additionally, we apply pruning at earlier stages. We stop the expansion as soon as we know that the resulting hypotheses would be pruned anyway. This is done in the function 'x isTooBadForCoverage C'. This way, we can avoid unnecessary computations and speed up the search significantly. Therefore, we store for each coverage C the score of its best lexical hypothesis. This score is updated whenever we generate a lexical hypothesis for coverage C with a better score. For this on-the-fly pruning to be effective, it is important that promising candidates are processed first. Therefore, we process

$A(C, \tilde{e}, j)$	maximizing argument of hypothesis $(C, \tilde{e}, j)$
$B(C, \tilde{e}, j)$	back pointer of hypothesis $(C, \tilde{e}, j)$
L_s	maximum source phrase length
$q_{\text{TM}}(j, j')$	best translation model score for translating source phrase $f_j, \ldots, f_{j'}$, i. e. $q_{\text{TM}}(j, j') = \max_{\tilde{e}} q_{\text{TM}}(\tilde{e}\|j, j')$
purgeCardinality c	free memory of cardinality c except trace back information
pruneCardinality c	apply coverage and cardinality pruning to all hypotheses with cardinality c
x isTooBadForCoverage C	check if score x cannot survive coverage or cardinality pruning

Figure 2: Notation and functions used in the search algorithm in Fig. 3.

```
    INPUT: source sentence $f_1^J$, translation options $E(j, j')$ for $1 \leq j \leq j' \leq J$, models $q_{\text{TM}}(\cdot)$, $q_{\text{LM}}(\cdot)$, $q_{\text{DM}}(\cdot)$
 0  $Q(\emptyset, \$, 0) = 0$ ; all other $Q(\cdot, \cdot, \cdot)$ entries are initialized to $-\infty$
 1  FOR cardinality $c = 1$ TO $J$ DO
 2      IF $c > L_s$ THEN purgeCardinality $c - L_s - 1$
 3      FOR source phrase length $l = 1$ TO $\min\{L_s, c\}$ DO
 4          FOR ALL coverages $C' \subset \{1, ..., J\} : |C'| = c - l$ DO
 5              FOR ALL start positions $j \in \{1, ..., J\} : C' \cap \{j, ..., j + l\} = \emptyset$ DO
 6                  coverage $C = C' \cup \{j, ..., j + l\}$
 7                  FOR ALL states $\tilde{e}', j' \in Q(C', \cdot, \cdot)$ DO
 8                      partial score $q = Q(C', \tilde{e}', j') + q_{\text{DM}}(j', j)$
 9                      IF $q + R(C, j + l) + q_{\text{TM}}(j, j + l)$ isTooBadForCoverage $C$ THEN CONTINUE
10                      FOR ALL phrase translations $\tilde{e}'' \in E(j, j + l)$ DO
11                          IF $q + R(C, j + l) + q_{\text{TM}}(\tilde{e}''|j, j + l)$ isTooBadForCoverage $C$ THEN BREAK
12                          score $= q + q_{\text{TM}}(\tilde{e}''|j, j + l) + q_{\text{LM}}(\tilde{e}''|\tilde{e}')$
13                          IF score$+R(C, j + l)$ isTooBadForCoverage $C$ THEN CONTINUE
14                          language model state $\tilde{e} = \tilde{e}' \oplus \tilde{e}''$
15                          IF score $> Q(C, \tilde{e}, j + l)$
16                          THEN $Q(C, \tilde{e}, j + l) =$ score
17                                $B(C, \tilde{e}, j + l) = (C', \tilde{e}', j')$
18                                $A(C, \tilde{e}, j + l) = \tilde{e}$
19      pruneCardinality $c$
```

Figure 3: DP beam search algorithm.

the coverage hypotheses in line 4 and the lexical hypotheses in line 7 in order of their scores, i. e. the best ones first. As already mentioned, the translation options $E(\cdot, \cdot)$ are sorted once before the search. In particular, we check the partial scores at the following points:

Line 9: at this point, we have accumulated the score of the predecessor hypothesis $Q(C', \tilde{e}', j')$, the distortion model score $q_{\text{DM}}(j', j)$, the rest score estimate of the new hypothesis and an optimistic estimate of the translation model score $q_{\text{TM}}(j, j + l)$. If this score is too bad for the coverage hypothesis C, there is no need to process any of the possible phrase translations.

Line 11: at this point, we have computed the score except for the LM. If this score is already too bad, there is no need to compute the LM score.

Line 13: at this point, the full score of the expansion is known. If we can prune a hypothesis here, we can directly reuse the memory and there is no need to check for recombination.

In Fig. 4, we show an illustration of the search. For each cardinality, we have a list of coverage hypotheses, here represented as boxes. For each coverage hypothesis, we have a list of lexical hypotheses, here represented as circles. We generate a specific lexical hypothesis (the filled circle) with cardinality c by expanding shorter hypotheses. The hypotheses with cardinality $c-1$ are expanded with one-word phrases, the hypotheses with cardinality $c - 2$ are expanded with two-word phrases etc. This differs from e. g. [8, 14] who expand hypotheses with cardinality c into higher cardinalities. Our approach has the advantage that all generated hypotheses have the same cardinality, which allows for more efficient recombination and pruning.

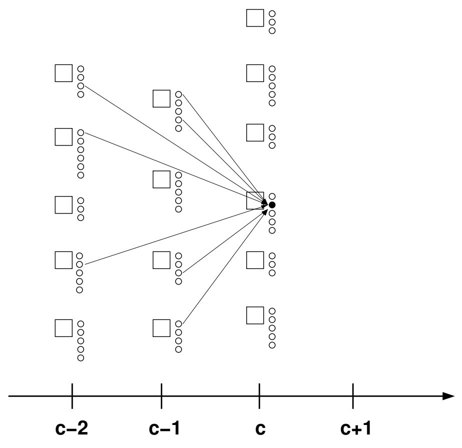

Figure 4: Illustration of the search. Hypotheses of cardinality c are generated by expanding shorter hypotheses. Boxes represent reordering hypotheses, circles represent lexical hypotheses.

Table 1: Statistics of the Chinese-English NIST task.

		Chinese	English
Train	Sentence pairs	8 M	
	Words	249 M	269 M
	Vocabulary size	251 K	431 K
Test	Sentences	878	3 512
	Words	25 K	105 K

6. Experimental results

The experiments were carried out on the large data track of the Chinese-English NIST task. The corpus statistics are shown in Tab. 1. The 4-gram LM was trained on about 650 M words. We use modified Kneser-Ney smoothing as implemented in the SRILM toolkit [15]. The model scaling factors were tuned to maximize the Bleu score. We report case-insensitive Bleu scores on the NIST 2002 evaluation set. The reordering limit in all experiments is 10 words.

Effect of search errors. In Fig. 5, we show the effect of the search errors. We varied the pruning parameters to generate translations of different quality. We show the Bleu score as a function of the model score $Q(\cdot)$ of the found translation. We observe a strong correlation between the model score and the Bleu score. Thus, it is important to find translation hypotheses with a good model score, i. e. a good search algorithm is important to achieve high translation quality.

Effect of rest score estimation. In Fig. 6, we show the effect of the rest score estimation (Sec. 4.2) on the translation performance. We compare the following variants of the rest score estimation. We varied the included models: translation model (TM), language model (LM),

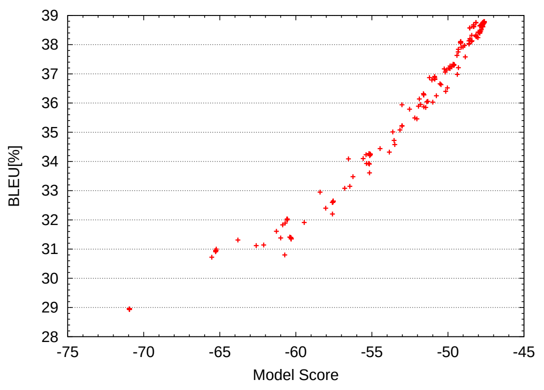

Figure 5: Effect of search errors. The larger the model score, the fewer search errors.

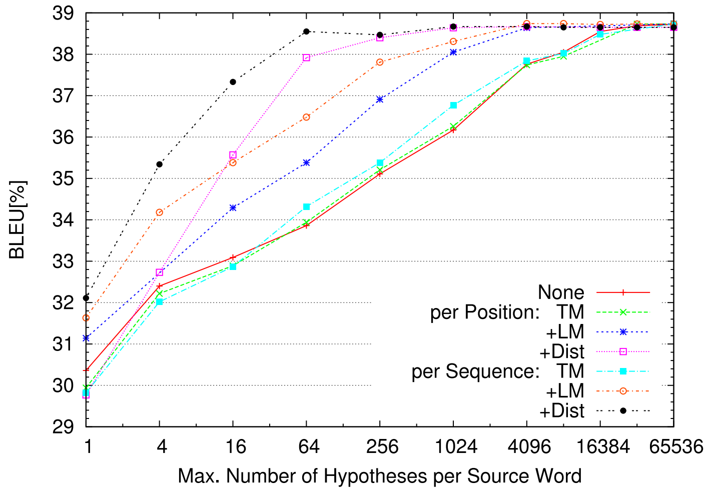

Figure 6: Effect of the rest score estimation on the translation performance.

distortion model (Dist) and we compare whether the rest scores are computed per position (Eq. 9) or per sequence (Eq. 7). For each of the rest score estimates, we varied the beam size. For (very) large beam sizes, the rest score estimation is of course not important. For small and medium beam sizes however, we observe that a good rest score estimate is important to achieve high Bleu scores. Using only the translation model for the rest score estimation does not help much. The curves are very similar to the one without rest score estimation. Both, the LM and the distortion model, help to improve the search results. The rest score estimates based on sequences of source position typically outperform the estimate based on positions, i. e. with the same beam size they achieve a higher Bleu score. The rest score estimate based on translation and LM scores for sequences of source positions in combination with a rest score estimate for the distortion penalty model works best. Using this rest score estimation, we already achieve a very

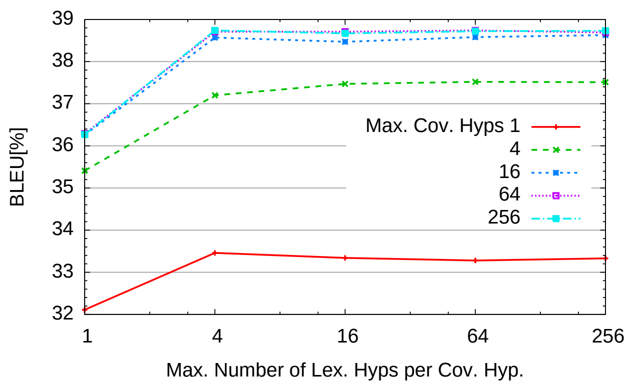

Figure 7: Effect of the number of lexical and coverage hypotheses.

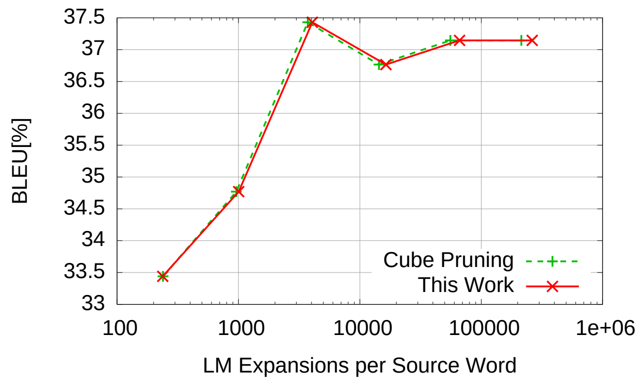

Figure 8: Comparison with cube pruning: translation quality as a function of the number of LM expansions.

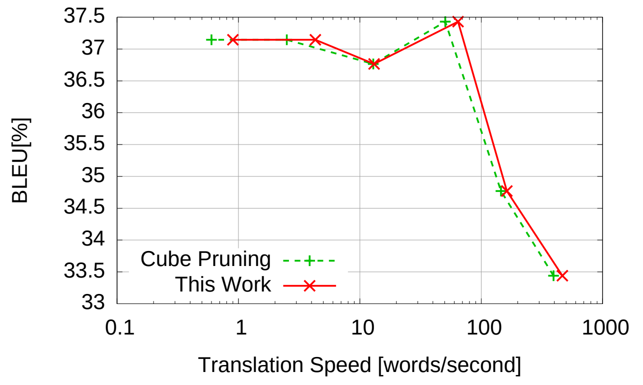

Figure 9: Comparison with cube pruning: translation quality as a function of the translation speed.

good Bleu score with a beam size of only 64 hypotheses. Without rest score estimation, we would need about 16 K hypotheses to achieve the same Bleu score.

Effect of lexical and coverage pruning. In Fig. 7, we separated the effect of lexical choice and reordering. For each curve, we limited the number of coverage hypotheses and varied the maximum number of lexical hypotheses per coverage hypothesis. Thus, along the x-axis we increase the search space by allowing for more lexical choice, whereas from curve to curve we allow for more reordering. To be precise: for each curve we fixed the histogram size for the coverage pruning per cardinality and we varied the histogram size for the lexical pruning. The overall search space is limited by the product of the two numbers; we vary the overall search space between 1 hypothesis and 64 K hypotheses. We observe that increasing the lexical choice beyond 16 hypotheses per coverage hypothesis does not lead to further improvements; often only 4 lexical hypotheses are sufficient. Furthermore, the improvement that achieved by taking more lexical alternatives into account is between 1 and 2 Bleu points. If we look at the maximum number of coverage hypotheses, we see a much bigger effect on the Bleu score. There is a considerable improvement by increasing the number of coverage hypotheses up to 64.

Comparison with cube pruning. We compared our implementation with cube pruning [16, 17]. We present the translation quality as a function of the number of LM expansions in Fig. 8. We observe that there is virtually no difference between cube pruning and our implementation. In Fig. 9, we compare the actual translation speed. Again, both systems are very similar. We conclude that the benefits of cube pruning, i. e. fewer LM expansions, can be achieved by simpler means as shown in this work. Note that we did not include the naive algorithm without on-the-fly pruning. This would have significantly more LM expansions and would be much slower, as shown in [17].

Comparison with Moses. We also performed a comparison with Moses [8], a publicly available decoder. We use the same TM, LM, model scaling factors etc. The lexicalized reordering model is not used in these experiments as the implementation in Moses differs slightly from our implementation; thus the results would not be comparable. One effect of this is that the results, i. e. Bleu scores, are somewhat worse than the ones reported in Fig. 7.

To compare the performance, we translated the test set with varying beam sizes and measured the Bleu score and the speed. The results are presented in Fig. 10. The beam size was varied between 1 hypothesis and 4 096 hypotheses, increasing by a factor of 4 from data point to data point. We observe that we can achieve the same Bleu score as Moses at a higher translation speed.

We are aware that comparisons of real time translation speed across implementations are problematic. We considered a comparison of the number of hypothesis

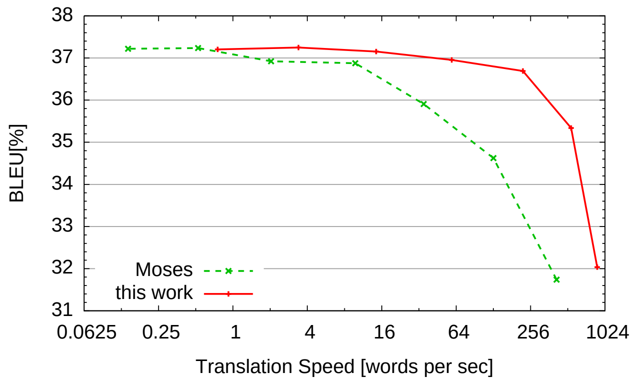

Figure 10: Comparison with Moses.

expansions, but we believe that a comparison of the speed in words per second is more interesting for the following reasons. First, a comparison of expansions is problematic as well, because an expansion might be more expensive in one implementation than in another. For instance with cube pruning, there is some overhead for maintaining the priority queues. Second, the overall goal is a fast decoder, i.e. to maximize the number of words per second, independent of the number of expansions that are performed. Therefore, we report the speed in words per second. The results indicate that the presented system is significantly more efficient than Moses.

7. Conclusions

We have described the search problem for phrase-based SMT in detail and presented an efficient solution. We showed that search errors deteriorate the translation results significantly, thus a good search algorithm is important. The analysis showed that it is important to focus the search on alternative coverage hypotheses. On the other hand, already a small number of lexical hypotheses per coverage is sufficient to achieve good translation quality. The presented method is competitive with cube pruning, yet much simpler to implement. A comparison with Moses showed, that the presented decoder is significantly faster at the same level of translation quality.

8. Acknowledgments

This material is partly based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023.

9. References

[1] F. J. Och, C. Tillmann, and H. Ney, "Improved alignment models for statistical machine translation," in *Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP)*, College Park, MD, June 1999, pp. 20–28.

[2] R. Zens, F. J. Och, and H. Ney, "Phrase-based statistical machine translation," in *25th German Conf. on Artificial Intelligence (KI2002)*, ser. Lecture Notes in Artificial Intelligence (LNAI), M. Jarke, J. Koehler, and G. Lakemeyer, Eds., vol. 2479. Aachen, Germany: Springer Verlag, September 2002, pp. 18–32.

[3] P. Koehn, F. J. Och, and D. Marcu, "Statistical phrase-based translation," in *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, Edmonton, Canada, May/June 2003, pp. 127–133.

[4] K. Knight, "Decoding complexity in word-replacement translation models," *Computational Linguistics*, vol. 25, no. 4, pp. 607–615, December 1999.

[5] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[6] F. Jelinek, *Statistical Methods for Speech Recognition*. Cambridge, MA: MIT Press, 1998.

[7] C. Tillmann and H. Ney, "Word reordering and a dynamic programming beam search algorithm for statistical machine translation," *Computational Linguistics*, vol. 29, no. 1, pp. 97–133, March 2003.

[8] P. Koehn and et al., "Moses: Open source toolkit for statistical machine translation," in *ACL Poster*, Prague, Czech Republic, June 2007, pp. 177–180.

[9] V. Steinbiss, B.-H. Tran, and H. Ney, "Improvements in beam search," in *Int. Conf. on Spoken Language Processing (ICSLP)*, September 1994, pp. 2143–2146.

[10] F. J. Och, "Statistical machine translation: From single-word models to alignment templates," Ph.D. dissertation, Lehrstuhl für Informatik 6, Computer Science Department, RWTH Aachen University, Aachen, Germany, October 2002.

[11] F. J. Och and H. Ney, "The alignment template approach to statistical machine translation," *Computational Linguistics*, vol. 30, no. 4, pp. 417–449, December 2004.

[12] P. Koehn, "Noun phrase translation," Ph.D. dissertation, University of Southern California, 2003.

[13] R. C. Moore and C. Quirk, "Faster beam-search decoding for phrasal statistical machine translation," in *MT Summit XI*, Copenhagen, Denmark, September 2007.

[14] C. Tillmann, "Efficient dynamic programming search algorithms for phrase-based SMT," in *Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, New York City, NY, June 2006, pp. 9–16.

[15] A. Stolcke, "SRILM – an extensible language modeling toolkit," in *ICSLP*, vol. 2, Denver, CO, September 2002, pp. 901–904.

[16] D. Chiang, "A hierarchical phrase-based model for statistical machine translation," in *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, Ann Arbor, Michigan, June 2005, pp. 263–270.

[17] L. Huang and D. Chiang, "Forest rescoring: Faster decoding with integrated language models," in *45th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, Prague, Czech Republic, June 2007.