A Greedy Decoder for Phrase-Based Statistical Machine Translation

Philippe Langlais, Alexandre Patry and Fabrizio Gotti
Département d'Informatique et de Recherche Opérationnelle
Université de Montréal,
C.P. 6128, succursale Centre-Ville
H3C 3J7, Montréal, Québec, Canada
{felipe,patryale,gottif}@iro.umontreal.ca

Abstract

Most phrase-based statistical machine translation decoders rely on a dynamic programming technique for maximizing a combination of models, including one or several language models and translation tables. One implication of this choice is the design of a scoring function that can be computed incrementally on partial translations, a restriction a search engine using a complete-state formulation does not have. In this paper, we present experiments we conducted with a simple, yet effective greedy search engine. In particular, we show that when seeded with the translations produced by a state-of-the-art beam search decoder, it produces an output of significantly higher quality than the latter taken alone, as measured by automatic metrics.

1 Introduction

At the beginning of Statistical Machine Translation (SMT), efforts were made to design efficient machine decoders for word-based models (Tillmann et al., 1997; Wang and Waibel, 1997; Niessen et al., 1998; García and Casacuberta, 2001). As phrase-based models gained in popularity (Koehn et al., 2003), specific phrase-based decoders were released, such as Pharaoh[1] (Koehn, 2004) and some open-source alternatives, among which Ramses (Patry et al., 2006) and Phramer (Olteanu et al., 2006).

All these decoders share one common property: they rely on a scoring function that is incremental, in order to allow an efficient organization of the computations by dynamic programming (DP). For the kind of models we typically consider in SMT (word- or phrase-based), this is just fine, but one can easily think of models for which such a property is inappropriate.

One notable exception to the dynamic programming approach is the ReWrite decoder (Germann et al., 2001). It is a greedy decoder that iteratively tries to improve a current translation by modifying some of its elements according to some predefined operations. At each iteration, the best hypothesis found up to that point is kept and used for the next iteration, until convergence is obtained, which typically happens after a few iterations. A time-efficient refinement of this decoder has been described in (Germann, 2003). However, Foster et al. (2003) did report that this decoder produces translations of lower quality than those produced by a DP-decoder.

To our knowledge, there has been no investigation on a greedy decoder designed to maximize the log-linear combination of models traditionally embedded in a phrase-based SMT system. This paper aims at filling this gap.

We show that our implementation, although not as good as a state-of-the-art beam search DP-engine, is not far off. More interestingly, we report experiments on the Europarl corpus where the greedy search algorithm significantly

[1] Moses, available at http://www.statmt.org/moses/ gracefully replaces Pharaoh.

improves the best translation produced by a DP-based decoder. Last, we demonstrate the flexibility of the approach by adding a reversed language model to the set of models consulted to score a translation.

The paper is organized as follows. We first describe our greedy search algorithm in Section 2. The experimental setup as well as the reference beam search DP-engine we used are described in Section 3. In Section 4, we report experiments comparing our greedy implementation with a state-of-the-art phase-based DP-search engine. We conclude our work in Section 5.

2 The greedy search engine

The strategy of `ReWrite`, as described in (Germann et al., 2001) is one of the simplest form of local search algorithms: a hill-climbing search. It uses a complete-state formulation, which means that it searches over the space of possible translations; while a typical beam search DP-decoder will typically explore the space of prefixes of all possible translations. Usually, a local search operates on a single state, which in our case defines the current translation and allows to move to neighboring states according to some predefined operations.

This local search strategy has three interesting characteristics. First, it requires a constant amount of memory, whereas a DP search requires an amount at the very least linear in the source sentence length. Second, it has been reported that local search algorithms indeed often propose a reasonable solution in combinatorial problems (Russell and Norvig, 1995). Third, the function we seek to optimize does not have to evaluate partial translations, a point we develop later on.

On the down side, the greedy search algorithm is obviously not optimal. In some situations, including ours, this is a risk we are willing to take.

The greedy search, which is sketched in Figure 1, depends on the definition of three functions: one that seeds the search with a current state (`seed`), a scoring function (`score`), which takes a candidate translation as an argument and that we seek to maximize, and a function (`neighborhood`), which returns a set of neighboring hypotheses to consider at each iteration.

Require: $source$ a sentence to translate
  $current \leftarrow$ `seed`($source$)
  loop
    $s\_current \leftarrow$ `score`($current$)
    $s \leftarrow s\_current$
    for all $h \in$ `neighborhood`($current$) **do**
      $c \leftarrow$ `score`(h)
      if $c > s$ **then**
        $s \leftarrow c$
        $best \leftarrow h$
    if $s = s\_current$ **then**
      return $current$
    else
      $current \leftarrow best$

Figure 1: Core of the greedy search algorithm.

2.1 The scoring function

In this study, we seek to maximize a log-linear combination of models typically used in state-of-the-art phrase-based DP-engines. In particular, in the first experiments we report, we maximize the very same function that `Pharaoh` maximizes and which is reported in Equation 1:

$$\begin{array}{rlll} Score(e,f) = & \lambda_{lm} & \log p_{lm}(f) & + \\ & \sum_i & \lambda_{tm}^{(i)} \log p_{tm}^{(i)}(f|e) & - \\ & \lambda_w & |f| & - \\ & \lambda_d & p_d(e,f) & \end{array} \quad (1)$$

where the λs are the weighting coefficients, p_{lm} is a language model, p_{tm}^i are different transfer tables (that share the same parameters in our experiments), $|f|$ stands for the length of the translation (counted in words), and $p_d(e,f)$ is a so-called distortion model (we used the simple one described in (Koehn et al., 2003)).

2.2 The neighborhood function

By inspecting translations produced by `Pharaoh`, we defined a set of six operations that can transform a current translation. This is by no means an exhaustive set, and extensions will be considered in future investigations. In particular, we do not yet allow words (or phrases) to be inserted or deleted, two operations that are used by the `ReWrite` decoder (Germann et al., 2001).

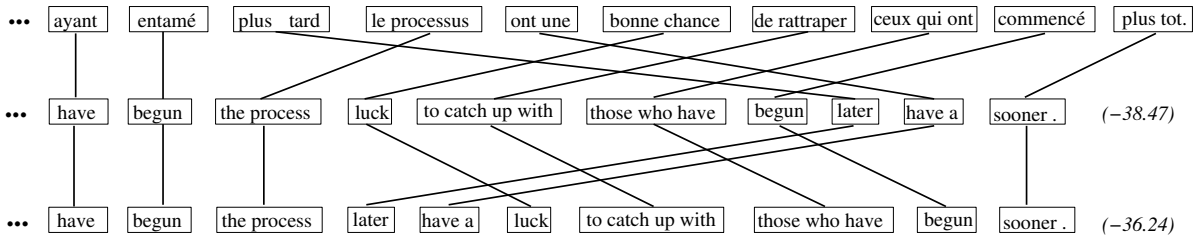

Figure 2: Illustration of an ill-formed translation produced by `Pharaoh` (second line) for an excerpt of a French sentence (first line). The third line shows the translation produced by `feGreedy` after one iteration.

Move The beam search DP-decoder tends to eliminate from the search space hypotheses that cover hard-to-translate segments. Since the decoder is forced to translate all the source material, it is often the case that the translation of those hard-to-translate segments is postponed until the very end of the search, typically producing ill-formed translations (see Figure 2). To overcome this situation to some extent, we allow some target phrases to move within the current translation.

Our implementation is very conservative: whenever two adjacent source phrases are translated by phrases that are distant,[2] we consider moving one of the translation closer to the other.

Swap It happens rather frequently that two adjacent source segments (words or phrases) do not form a phrase that belongs to the transfer table. The order in which their respective translations will be output will be strongly influenced by the compromise between the possible inversions the language model allows and the strong bias toward monotonous translations the distortion model has. For this reason, we defined an operation which allows to swap two adjacent target segments. The complexity of this operation[3] is $\mathcal{O}(N-1)$, that is, linear in the number N of source phrases in the current hypothesis.

Replace This operation simply allows to change the translation given for a specific source segment by another one found in the transfer table. This operation has a complexity of $\mathcal{O}(N \times T)$, where T is the maximum number of translations considered per source phrase.[4]

Bi-replace With the same idea in mind, we allow the translation of two adjacent source phrases to change simultaneously. We hope that by changing more than one unit, the search will likely escape a local maximum. The complexity of this operation is $\mathcal{O}(T^2 \times (N-1))$, that is, quadratic in the number of translations considered per source phrase.

Split One task a beam search DP-decoder handles—most of the time implicitly—is the segmentation of the source material into phrases. We allow our decoder to split in two parts a given source phrase. While doing so, the two new source phrases receive a translation found in the transfer table (we consider all of them). The complexity of this operation is $\mathcal{O}(N \times S \times T^2)$, where S is the (average) number of words a source phrase has in the current hypothesis.

Merge As opposed to the `split` operation, the merge operation allows two adjacent source phrases to be merged, in which case a new translation is also picked from the translation table. This operation is $\mathcal{O}(T \times (N-1))$.

2.3 The seed translation

2.3.1 From scratch

In `ReWrite`, the seed translation is formed by collecting for each word its best translation as provided by the transfer table. This is the idea we implemented as well. There is one subtlety however, when we deal with phrases: a segmentation of the source sentence S into phrases must

[2] As defined by a threshold value counted in words. We used 3 in our experiments.

[3] We measure complexity here in terms of the maximum number of hypotheses that will be considered, given a current one.

[4] A typical value of T in our experiments is 10.

be performed. Since many source phrases overlap, there are many a priori segmentations we can choose from. In our case, we select the segmentation which involves the minimum number of source phrases belonging to the translation model $\mathcal{M}$ that cover maximally the source sentence S.

To do so, it suffices to consider $\mathcal{M}$ as a set of *spans* $\langle i, j \rangle$ denoting the fact that a source phrase in $\mathcal{M}$ covers the positions i to j (counted in words) in S. We define an item τ_s as a triple $\langle b, c, n \rangle$ which respectively stores b, the beginning of a span $\langle b, s \rangle$ ending in position s; c, the number of source words covered so far, and n, the number of source phrases used to cover S up to position s. Intuitively, an item τ_s stores the best coverage found so far from the beginning of the source sentence to position s, along with the number of source phrases used so far.

We compute the item $\tau_{|S|}$ by the recursion described in Equation 2, where we define for an item $\tau \equiv \langle b, c, n \rangle$, the operators $b(\tau)$, $c(\tau)$ and $n(\tau)$ to be respectively b, c and n.

$$\tau_s = \max \left\{ \begin{array}{l} \langle 0, 0, 0 \rangle, \\ \max\limits_{\substack{d \leq s \ : \\ \langle d, s \rangle \in \mathcal{M}}} \left\{ \begin{array}{l} \langle d, \\ c(\tau_d) + s - d + 1, \\ n(\tau_d) + 1 \qquad \rangle \end{array} \right\} \end{array} \right\} \quad (2)$$

The maximizations involved in Equation 2 are carried out over a set of items. We use the following operator to compare two items:

$$\max\{\tau_1, \tau_2\} = \left\{ \begin{array}{ll} \tau_2 & \text{if} \left\{ \begin{array}{ll} c(\tau_1) < c(\tau_2) & \text{or} \\ c(\tau_1) = c(\tau_2) & \text{and } n(\tau_1) > n(\tau_2) \end{array} \right. \\ \tau_1 & \text{otherwise} \end{array} \right. \quad (3)$$

The coverage is obtained by simply backtracking from item $\tau_{|S|}$, that is, by computing the set $\beta\tau_{|S|}$:

$$\beta_{\tau_e} = \left\{ \begin{array}{ll} \phi & \text{if } e \equiv 0 \\ \{\langle b(\tau_e), e \rangle\} \cup \beta(\tau_{\delta(e)}) & \text{otherwise} \end{array} \right. \quad (4)$$
$$\text{with } \delta(e) = \text{argmax}_{b<e}\ c(\tau_b) \neq 0$$

The recursion involved in this computation lends itself to an efficient computation by dynamic programming. Once the source segmentation is found, we simply pick for each source phrase the best translation found in $\mathcal{M}$. An illustration of the segmentation obtained for a short source sentence is provided in Figure 3.

2.3.2 Seeding `feGreedy` with `Pharaoh`

It is likely that a DP-search will outperform our greedy implementation, hereafter named `feGreedy`. Therefore, it is natural to investigate whether any benefit would result from seeding `feGreedy` with the best translation produced by `Pharaoh`.[5]

The idea of cascading two translation engines has been pioneered within the word-based Candide translation system (Berger et al., 1994). Unfortunately, the authors did not describe their local search engine, neither did they provide an evaluation of its benefits to the overall system. The cascading strategy received a more dedicated treatment in Marcu (2001) and Watanabe and Sumita (2003). In their work, the authors were seeding a word-based greedy search algorithm with examples extracted from a translation memory hoping to bias the search toward a better solution. Our motivation is slightly different however: we simply want to know whether the greedy strategy can overcome some search errors made by a phrase-based DP-search.

3 Experimental setup

3.1 Corpora

We concentrated our efforts on the shared task of last year's workshop on Statistical Machine Translation (Koehn and Monz, 2006) which consisted in translating Spanish, German and French texts into English and the reverse direction. The training material available is coming from the Europarl corpus. Four disjoint corpora were released during this exercise, namely `train`, a portion of 688,031, 730,740 and 751,088 pairs of sentences for French, Spanish and German respectively; `dev`, a development corpus that we used for tuning; `devtest`, a corpus of 2,000 pairs of

[5] We used the `--trace` option of `Pharaoh` to access the phrasal alignment corresponding to the best translation found.

F	de plus , nos systèmes administratifs doivent être modernisés . nous devons également donner le bon exemple .
E	in addition , our administrative systems must be modernised , and it is our duty to lead by example .
S_0	[de plus ,] [nos systèmes administratifs] [doivent] [être modernisés] [. nous devons également] [donner le bon exemple .]
T_0	[furthermore ,] [our administrative systems] [must] [modernization] [and we also need] [set a good example .] *-19.5068*
S_1	[de plus ,] [nos systèmes administratifs] [doivent] [être modernisés] **[.] [nous devons également]** [donner le bon exemple .]
T_1	[furthermore ,] [our administrative systems] [must] [modernization] **[.] [we must also]** [set a good example .] SPLIT *-17.4382*
S_2	[de plus ,] [nos systèmes administratifs] [doivent] **[être] [modernisés]** [.] [nous devons également] [donner le bon exemple .]
T_2	[furthermore ,] [our administrative systems] [must] **[be] [modernized]** [.] [we must also] [set a good example .] SPLIT *-15.8488*
S_3	[de plus ,] [nos systèmes administratifs] [doivent] [être] [modernisés] [.] [nous devons également] **[donner] [le bon exemple .]**
T_3	[furthermore ,] [our administrative systems] [must] [be] [modernized] [.] [we must also] **[give] [a good example .]** SPLIT *-15.5885*
S_4	**[de plus ,]** [nos systèmes administratifs] [doivent] [être] [modernisés] [.] [nous devons également] [donner] [le bon exemple .]
T_4	**[in addition ,]** [our administrative systems] [must] [be] [modernized] [.] [we must also] [give] [a good example .] REPLACE *-15.5199*

Figure 3: Steps involved by the translation of a French sentence (F); E is its reference translation. A segmentation (S_0) is first chosen from the 49 different source phrases that cover partially F. T_0 is the associated seed translation. The phrases in bold are those involved in the highest-scored operation at each iteration. Over 4,100 hypotheses have been evaluated within a time period of 300 milliseconds.

sentences that we used for monitoring our system; and `test`, the official test set of the 2006 shared task, that we used only for final tests. We further split the `test` corpus in two parts, `test-in`, the in-domain part which consists of 2,000 sentences from the European parliament debates, and `test-out`, which counts 1,034 sentences[6] collected from editorials of the Project Syndicate website.

3.2 Phrase-based engine

The reference system we used for comparison purposes is the state-of-the-art phrase-based engine which was made available by the organizers of the shared task. The language model (a trigram) was trained using the SRILM toolkit (Stolcke, 2002), and the translation tables (phrases up to 7 words long) were obtained by running the scripts provided. These tables contain 4 scores (relative frequencies and lexical scores in both direction) that each receives a weighting coefficient. A fifth score is intended to serve as a phrase penalty model. The `Pharaoh` built-in distortion model and a word penalty component receive as well a weighting coefficient. Altogether, 8 coefficients were tuned using the script `minimum-error-rate-training.perl`.

For most of our experiments, the threshold values that `Pharaoh` uses were left to their built-in defaults. This is the version of our DP-system that we call BASE from now on.

[6] We removed 30 sentences with encoding problems.

Systems	L	en→L WER	en→L BLEU	L→en WER	L→en BLEU
BASE	fr	55.12	30.16	51.47	29.23
G-S	fr	54.27	29.83	50.93	29.13
G-BASE	fr	**53.62**	**30.64**	**50.37**	**29.62**
BASE	es	55.04	28.17	50.97	29.94
G-S	es	**53.22**	28.99	50.77	29.67
G-BASE	es	**53.14**	**28.72**	**50.04**	**30.30**
BASE	de	62.38	17.32	60.12	24.54
G-S	de	62.53	17.03	**57.55**	23.84
G-BASE	de	**61.85**	**17.51**	**58.33**	24.97

Table 1: Performances of different search algorithms measured on the `devtest` corpus, as a function of the translation direction. The figures in bold are significantly better than the corresponding BASE configuration at the 99% confidence level.

4 Experiments

4.1 `feGreedy` with or without `Pharaoh`

We first compared `feGreedy` with BASE by running both decoders, with the same function to maximize (see Equation 1). In one version of the greedy search, G-S, the search was initiated from scratch (see Section 2.3.1). In a second version, G-BASE, the search was seeded with the best translation produced by `Pharaoh` (see Section 2.3.2). The results are reported in Table 1.

Expectedly, for all translation directions, the greedy search algorithm alone provides translations of significantly lower quality than the DP-search. This is consistent with the observations made by (Foster et al., 2003) in word-based translation experiments. However, we observe that the greedy search improves upon the best translation that BASE found. This seems to be consistent for all translation directions and for both evaluation metrics considered. For all translation directions except German-to-English, the improvements are significant at the 99% confidence level.[7]

In order to better appreciate the situation, we report in Table 2 more specific information on what the greedy search accomplishes. We restrict ourselves to translating into English, since this corresponds to the most studied translation direction in the SMT literature, and we did not notice clear differences in the reverse direction.

First of all, we observe that roughly 40% of the translations produced by BASE get improved in score (Equation1) by `feGreedy`. We were expecting a much lower improvement proportion. One explanation for that might be the stack-size limit `Pharaoh` considers as a default (100). Keeping the first hundred best hypotheses for each source coverage (i.e. the number of source words covered by a given hypothesis) might bias the search toward locally optimal hypotheses. More expectedly, however, we observe that more than 90% of the seed translations computed by the technique described in Section 2.3.1 get improved by `feGreedy`.

Regarding the selected operations at each iteration, roughly 40% of them are replacement ones, that is, the replacement of one translation by another one. The `move` operation also highly beneficial. The fact that more than 15% of the winning operations in G-BASE are split operations might appear surprising at first. Recall that this operation comes along with a possible change in the target material and is therefore not just a matter of segmenting differently the source material. We also observe that some operations are only marginally useful. This is the case of `merge` and `swap`. The fact that the `swap` operation is not productive just indicates that the phrase table is already doing a good job at capturing local word-order differences. We do not have yet a clear explanation for the low impact of the `merge` operation.

Last, we can see from Table 2 that the distribution of the number of iterations required by G-BASE and G-S are very different. The former configuration requires only a few iterations to converge: at most 2 iterations in approximatively 70% of the cases. For the latter, only more than half of the translations are completed after 4 iterations. Both versions require less than 10 iterations on average to produce a translation.

It is worthwhile to note that, although we did not yet pay attention to translation speed within our current implementation,[8] `feGreedy` defi-

[7] In all our experiments, we used the bootstrap resampling method described in (Zhang and Vogel, 2004) to compute significance levels, evaluating 1,000 samplings of 700 sentences each.

[8] It is a simple matter to improve the speed of

	fr→en		es→en		de→en	
	G-B	G-S	G-B	G-S	G-B	G-S
%up	42.6	93.5	37.1	90.8	42	95.8
↑ log-s	3.6	2.9	2.7	1.7	1.8	2.9
$\%it. < 2$	44.6	13.5	50.7	13.8	43.1	6.5
$\%it. < 3$	66.2	29.7	74.4	31.6	65.7	17.2
$\%it. < 5$	90.8	59.7	93.3	65.7	91.7	45.0
$\%it. < 10$	98.8	95.0	100.0	97.8	100.0	87.5
MOVE	42.2	–	44.0	–	42.1	–
REPLACE	41.3	45.1	38.3	45.3	37.7	51.7
SPLIT	14.9	52.8	16.3	52.4	18.6	46.5
MERGE	0.9	1.7	0.8	1.8	1.0	1.1
SWAP	0.5	0.2	0.2	0.2	0.3	0.5

Table 2: Profile of two variants of `feGreedy` on the `devtest` corpus. G-B is a shorthand for G-BASE. %up stands for the percentage of sentences that get improved by the greedy search. ↑ log-s indicates the average gain in score (Equation 1). $it. < n$ indicates the percentage of sentences improved for which less than n iterations were required. The bottom part of the table indicates the percentage of operations that ranked best at an iteration of the greedy search.

nitely compares favorably to BASE in that respect. Currently, translating the 1,000 sentences of `devtest` on a Pentium computer clocked at 3 GHz requires 9 minutes with `feGreedy`, compared to 78 minutes with BASE.

4.2 Further experimenting with `feGreedy`

In the previous section, we conducted a pairwise comparison of `feGreedy` with our reference system, by providing the greedy decoder the same function `Pharaoh` is maximizing. In this section, we report experiments we conducted in order to improve `feGreedy`. Our starting point is the configuration of the greedy search seeded with the best translation produced by BASE.

4.2.1 Adding new features

One strength of the greedy search is that it operates on a full candidate translation. This allows us to optimize a scoring function which is not necessarily incremental. To illustrate this added flexibility, we added a reversed n-gram language model to the set of models of the scoring function maximized by `Pharaoh`. We call this variant G-REV.

A reversed n-gram model simply predicts each word of the translation from right to left, as described in Equation 5. At first glance, this might seem like an odd thing to do, since there is probably not much information a decoder can gain from this model. Yet, this is one of the simplest models imaginable, which could not be easily integrated into a DP-decoder such as `Pharaoh`, since the suffix of a hypothesis is unknown during the search.

$$p(t_1^T) \approx \prod_{i=1}^{T} p(t_i | t_{i+1} \dots t_{i+n-1}) \qquad (5)$$

Because we added a new model to the linear combination optimized by `feGreedy`, we had to tune the coefficients involved once more. To save some computation time, however, we did not explore the full range of values for each coefficient, but concentrated on values close enough to those we had already found. The results of this experiment are reported in Table 3.

For all translation directions but Spanish-to-English, the gain in performance, as measured by WER, are very small if not negative. However, improvements in BLEU, although mostly not significant, are consistent for all translation directions.

		en→L		L→en	
Systems	L	WER	BLEU	WER	BLEU
BASE	fr	55.12	30.16	51.47	29.23
G-BASE	fr	53.62	30.64	50.37	29.62
G-REV	fr	53.65	30.85	50.30	29.70
BASE	es	55.04	28.17	50.97	29.94
G-BASE	es	53.14	28.72	50.04	30.30
G-REV	es	52.37	29.31	50.05	30.33
BASE	de	62.38	17.32	60.12	24.54
G-BASE	de	61.85	17.51	58.33	24.97
G-REV	de	61.85	17.57	57.99	25.20

Table 3: Performances of the G-REV variant for different translation directions, measured on the `devtest` corpus.

`feGreedy`, since in our current implementation, any operation applied to a hypothesis triggers the computation of its score from scratch, while some straightforward bookkeeping would eliminate most of the computations.

4.2.2 A beam-search version of `feGreedy`

As we already noted, one advantage of the greedy search is that it requires a set amount of memory, since it does not build a search graph like DP-search engines do (e.g. `Pharaoh`). This is an interesting advantage, but keeping only a single-best current translation is somehow too heavy-handed a response to the memory problem. Therefore, in this experiment, we tested a variant of the greedy search, technically known as local beam search (Russell and Norvig, 1995). In this greedy search, a beam of at most k best hypotheses are kept at each iteration. The search tries to improve on each of them, until no improvement can be found. We call this version G-BEAM.

We populate the beam with k seed hypotheses. One is the best translation proposed by BASE, as described in section 2.3.2. The $k-1$ others are derived from the source coverage we compute, as described in Section 2.3.1. To form the ith seed translation, we select the ith-best translation of each source phrase, as found in the transfer table. Obviously, they are many other ways we could proceed to produce k seed translations, including considering the k-first hypotheses produced by BASE. An example of seed translations produced for one short sentence is reported in Figure 4. In this example, as is often the case, the seed hypothesis proposed by BASE is ranked higher than the one computed from scratch.

No improvement in BLEU and WER have been observed over the 1-best greedy search seeded with `Pharaoh` (G-BASE). This is disappointing, but not entirely surprising, since BASE already does a good job, and that G-BASE further improved on it. What is more interesting, however, is that the beam version of our greedy search managed to find higher-scored translations (according to Equation 1) than G-BASE does. On one hand, this is satisfactory from a search point of view. On the other hand, it is disturbing to note that search errors are at some point beneficial! The adequacy of the evaluation metrics we considered might be one reason for this observation. However, we believe that the problem is more likely due to severe (well-known) shortcomings of the scoring function we seek to maximize, including its blindness to syntactical quality.

Averaged across all translation directions, roughly 20% of the translations produced by G-BEAM are different from those produced by G-BASE. Among these modified translations, 87% have a higher score (Equation 1). The fact that for some sentences, G-BEAM missed an optimum found by G-BASE is simply due to the greediness of the search along with a limited beam size. We observed that by increasing the beam size, the number of downgraded translations produced by G-BEAM decreases. By simply choosing the best-scored translation produced by either G-BASE or G-BEAM, we did not manage to improve significantly BLEU and WER figures.

cette question est , bien sûr , parfaitement légitime , mais il faut y répondre de façon correcte et précise . *(source sentence)*
◇ this question is , of course , perfectly legitimate , but it must be answered properly and carefully. (`Pharaoh`, *-16.11*)
◇ subject is of course , perfectly legitimate , but we must respond to properly and carefully. (scratch-1, *-18.22*)
◇ subject is of course fully justified , but it must be answered properly and carefully. (scratch-3, *-20.58*)
◇ subject is of course perfectly quite legitimate , but it must be answered properly and carefully. (scratch-2, *-21.57*)

Figure 4: 4 seed translations computed for the source (French) sentence at the top along with their score. *scratch-n* stands for a seed translation computed from scratch, picking for each source phrase belonging to the coverage, the nth translation found in the transfer table.

4.2.3 Final tests

We conclude our exploration of `feGreedy` by running on the `test` corpus the most salient versions we tested on the development corpus: BASE, the `Pharaoh` DP-decoder, G-BASE, the greedy search engine seeded with the best translation BASE found, G-BEAM-5, the local beam variant of `feGreedy`, with a beam size of 5, and G-REV, the greedy variant using a reversed language model.

Results are reported in Table 4 and 5 for the in- and out-domain test material respectively.

Systems	L	en→L WER	en→L BLEU	L→en WER	L→en BLEU
BASE	fr	54.85	30.90	51.69	29.96
G-BASE	fr	**53.38**	**31.42**	**50.46**	**30.27**
G-BEAM-5	fr	**53.46**	31.26	**50.40**	30.13
G+B5	fr	**53.43**	**31.28**	**50.36**	30.17
G-REV	fr	**53.49**	**31.52**	**50.48**	**30.25**
BASE	es	54.23	29.64	51.04	30.54
G-BASE	es	**52.77**	**30.14**	**50.02**	**30.87**
G-BEAM-5	es	**52.61**	30.24	**50.12**	**30.89**
G+B5	es	**52.61**	**30.25**	**50.11**	**30.93**
G-REV	es	**52.67**	29.79	**50.07**	30.84
BASE	de	62.32	17.68	60.54	24.45
G-BASE	de	**61.73**	17.88	**58.85**	24.66
G-BEAM-5	de	61.98	17.82	**57.62**	24.59
G+B5	de	61.95	17.84	**57.62**	24.58
G-REV	de	**61.77**	17.89	**58.48**	24.82

Table 4: Performances of different search algorithms measured on the `test-in` corpus. Figures in bold are significantly better than their BASE counterpart at the 99% confidence level.

Systems	L	en→L WER	en→L BLEU	L→en WER	L→en BLEU
BASE	fr	60.29	22.31	56.66	20.78
G-BASE	fr	**57.80**	**23.44**	**54.70**	**21.38**
G-BEAM-5	fr	**57.68**	**22.91**	**54.44**	**21.28**
G+B5	fr	**57.61**	**23.03**	**54.43**	**21.33**
G-REV	fr	**58.12**	**23.25**	**54.66**	**21.37**
BASE	es	57.07	24.20	51.11	25.17
G-BASE	es	**54.83**	**25.09**	**49.77**	**25.59**
G-BEAM-5	es	**54.16**	24.91	**49.74**	**25.74**
G+B5	es	**54.11**	**24.95**	**49.72**	**25.69**
G-REV	es	**53.46**	**26.33**	**49.80**	**25.64**
BASE	de	67.09	11.00	65.62	16.00
G-BASE	de	**65.79**	**11.49**	**63.51**	16.38
G-BEAM-5	de	**66.12**	11.24	**61.54**	**16.72**
G+B5	de	**66.10**	11.33	**61.53**	**16.74**
G-REV	de	**65.93**	11.40	**62.96**	16.38

Table 5: Performances of different search algorithms measured on the `test-out` corpus. Figures in bold are significantly better than their BASE counterpart at the 99% confidence level.

First, we observe that the greedy variant G-BASE outperforms the BASE algorithm, for both in- and out-domain. The improvements in WER and BLEU are significant (at the 99% confidence level) for all translation directions, but German-to-English. This is consistent with our previous experiments on the development corpus.

Second, the beam version of `feGreedy`, although significantly better than BASE in most cases, performs usually marginally worse than the corresponding G-BASE variant. The observation we made on the development corpus still holds: the beam variant of the search manages to find translations that are better scored by Equation 1. On the out-domain (resp. in-domain) corpus, 34% (resp. 17%) of the translations produced by G-BEAM-5 did improve in score compared with their G-BASE counterpart. Less than 4% (resp. 3%) received a lower score. The fact that, on the out-domain corpus, twice as many translations receive an higher score with the beam version is encouraging, even if it does not clearly pay off in terms of evaluation metrics.

Picking the highest-scored translation (Equation 1) proposed by either G-BASE or G-BEAM-5 slightly improves upon the G-BEAM-5 variant for almost all translation directions, but the gain is not significant. The corresponding figures are reported as the G+B5 variant in Tables 4 and 5.

5 Conclusions

In this study, we addressed the problem of searching the space of possible translations with a greedy search algorithm designed to maximize the log-linear function many state-of-the-art phrase-based systems use. We discussed some advantages of search algorithms working on a complete-state representation as our greedy search does. We conducted experiments showing that it could improve the best translation found by the more demanding multi-stack beam-search dynamic-programming algorithm embedded in decoders such as `Pharaoh` or `Ramses`.

Perhaps the main contribution of this study is to point out the potential such an easy search algorithm has over more demanding decoders. Until now, this was an idea that had not received much attention in the phrase-based SMT community.

We plan to extend this work in several directions. Actually, one initial motivation for this

study was to explore post-processing operations that could apply to the output of a translation engine, in order to recover systematic errors, in a way inspired by transformation-based learning (Brill, 1995). On step toward accomplishing this consists in increasing the number of operations that our greedy search can perform, associating with each of them a coefficient that we can adjust on a development corpus. This is the idea we want to explore further.

We also want to cast our greedy decoder within the open-source framework called `Mood`, whose principle is to offer decoders that are easy to modify and extend. Therefore, our goal will be to release a reengineered version of `feGreedy`.

6 Acknowledgements

This study has been partially funded by a NSERC grant. We are grateful to Pierre Poulin for his fruitful comments on this work.

References

Adam L. Berger, Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, John R. Gillett, John D. Lafferty, Robert L. Mercer, Harry Printz, and Luboš Ureš. 1994. The Candide system for machine translation. In *Proceedings of HLT*, pages 157–162, Morristown, NJ, USA.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

George Foster, Simona Gandrabur, Philippe Langlais, Pierre Plamondon, Graham Russel, and Michel Simard. 2003. Statistical machine translation: Rapid development with limited resources. In *MT Summit IX*, pages 110–117, New Orleans.

Ismael García and Francisco Casacuberta. 2001. Search algorithms for statistical machine translation based on dynamic programming and pruning techniques. In *Proceedings of the 8th MT Summit*, pages 115–120, Santiago de Compostela, Spain.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting of the ACL*, pages 228–235, Toulouse, France.

Ulrich Germann. 2003. Greedy decoding for statistical machine translation in almost linear time. In *HLT-NAACL*, pages 72–79, Edmonton, Canada.

Philipp Koehn and Christof Monz. 2006. Manual and automatic evaluation of machine translation between European languages. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, pages 102–121, New York City, June.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT*, pages 127–133.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based SMT. In *Proc. of the 6th AMTA*, pages 115–124, Washington, DC.

Daniel Marcu. 2001. Towards a unified approach to memory- and statistical-based machine translation. In *Proceedings of the 39th Annual Meeting of the ACL*, pages 378–385, Toulouse, France.

Sonia Niessen, Stephen Vogel, Hermann Ney, and Christof Tillmann. 1998. A DP-based search algorithm for statistical machine translation. In *Proceedings of the 36th Annual Meeting of the ACL and 17th COLING*, pages 960–966, Montréal, Canada.

Marian Olteanu, Chris Davis, Ionut Volosen, and Dan Moldovan. 2006. Phramer – an open source statistical phrased-based translator. In *Proceedings of the HLT/NAACL Workshop on Statistical Machine Translation*, pages 150–153, New York, USA.

Alexandre Patry, Fabrizio Gotti, and Philippe Langlais. 2006. Mood: A modular object-oriented decoder for statistical machine translation. In *5th LREC*, pages 709–714, Genoa, Italy, May.

Stuart Russell and Peter Norvig. 1995. *Artificial Intelligence. A Modern Approach*. Prentice Hall.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of ICSLP*, Denver, Colorado, Sept.

Christoph Tillmann, Stephen Vogel, Hermann Ney, and A. Zubiaga. 1997. A DP-based search using monotone alignments in statistical translation. In *Proceedings of the 35th Annual Meeting of the ACL*, pages 289–296, Madrid, Spain.

Ye-Yi Wang and Alex Waibel. 1997. Decoding algorithm in statistical machine translation. In *Proceedings of the 35th Annual Meeting of the ACL*, pages 366–372, Madrid, Spain.

Taro Watanabe and Eiichiro Sumita. 2003. Example-based decoding for statistical machine translation. In *Machine Translation Summit IX*, pages 410–417, New Orleans, Louisiana.

Ying Zhang and Stephan Vogel. 2004. Measuring confidence intervals for the machine translation evaluation metrics. In *Proceedings of the 10th TMI*, pages 85–94, Baltimore, Maryland, USA.