Document-Wide Decoding for Phrase-Based Statistical Machine Translation

Christian Hardmeier Joakim Nivre Jörg Tiedemann
Uppsala University
Department of Linguistics and Philology
Box 635, 751 26 Uppsala, Sweden
firstname.lastname@lingfil.uu.se

Abstract

Independence between sentences is an assumption deeply entrenched in the models and algorithms used for statistical machine translation (SMT), particularly in the popular dynamic programming beam search decoding algorithm. This restriction is an obstacle to research on more sophisticated discourse-level models for SMT. We propose a stochastic local search decoding method for phrase-based SMT, which permits free document-wide dependencies in the models. We explore the stability and the search parameters of this method and demonstrate that it can be successfully used to optimise a document-level semantic language model.

1 Motivation

In the field of translation studies, it is undisputed that discourse-wide context must be considered carefully for good translation results (Hatim and Mason, 1990). By contrast, the state of the art in statistical machine translation (SMT), despite significant advances in the last twenty years, still assumes that texts can be translated sentence by sentence under strict independence assumptions, even though it is well known that certain linguistic phenomena such as pronominal anaphora cannot be translated correctly without referring to extra-sentential context. This is true both for the phrase-based and the syntax-based approach to SMT. In the rest of this paper, we shall concentrate on phrase-based SMT.

One reason why it is difficult to experiment with document-wide models for phrase-based SMT is that the dynamic programming (DP) algorithm which has been used almost exclusively for decoding SMT models in the recent literature has very strong assumptions of locality built into it. DP beam search for phrase-based SMT was described by Koehn et al. (2003), extending earlier work on word-based SMT (Tillmann et al., 1997; Och et al., 2001; Tillmann and Ney, 2003). This algorithm constructs output sentences by starting with an empty hypothesis and adding output words at the end until translations for all source words have been generated. The core models of phrase-based SMT, in particular the n-gram language model (LM), only depend on a constant number of output words to the left of the word being generated. This fact is exploited by the search algorithm with a DP technique called *hypothesis recombination* (Och et al., 2001), which permits the elimination of hypotheses from the search space if they coincide in a certain number of final words with a better hypothesis and no future expansion can possibly invert the relative ranking of the two hypotheses under the given models. Hypothesis recombination achieves a substantial reduction of the search space without affecting search optimality and makes it possible to use aggressive pruning techniques for fast search while still obtaining good results.

The downside of this otherwise excellent approach is that it only works well with models that have a local dependency structure similar to that of an n-gram language model, so they only depend on a small context window for each target word. Sentence-local models with longer dependencies can be added, but doing so greatly increases the risk for search errors by inhibiting hypothesis recombination. Cross-sentence dependencies cannot be directly integrated into DP SMT decoding in

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1179–1190, Jeju Island, Korea, 12–14 July 2012.

any obvious way, especially if joint optimisation of a number of interdependent decisions over an entire document is required. Research into models with a more varied, non-local dependency structure is to some extent stifled by the difficulty of decoding such models effectively, as can be seen by the problems some researchers encountered when they attempted to solve discourse-level problems. Consider, for instance, the work on cache-based language models by Tiedemann (2010) and Gong et al. (2011), where error propagation was a serious issue, or the works on pronominal anaphora by Le Nagard and Koehn (2010), who implemented cross-sentence dependencies with an ad-hoc two-pass decoding strategy, and Hardmeier and Federico (2010) with the use of an external decoder driver to manage backward-only dependencies between sentences.

In this paper, we present a method for decoding complete documents in phrase-based SMT. Our decoder uses a local search approach whose state consists of a complete translation of an entire document at any time. The initial state is improved by the application of a series of operations using a hill climbing strategy to find a (local) maximum of the score function. This setup gives us complete freedom to define scoring functions over the entire document. Moreover, by optionally initialising the state with the output of a traditional DP decoder, we can ensure that the final hypothesis is no worse than what would have been found by DP search alone. We start by describing the decoding algorithm and the state operations used by our decoder, then we present empirical results demonstrating the effectiveness of our approach and its usability with a document-level semantic language model, and finally we discuss some related work.

2 SMT Decoding by Hill Climbing

In this section, we formally describe the phrase-based SMT model implemented by our decoder as well as the decoding algorithm we use.

2.1 SMT Model

Our decoder is based on local search, so its state at any time is a representation of a complete translation of the entire document. Even though the decoder operates at the document level, it is important to keep track of sentence boundaries, and the individual operations that are applied to the state are still confined to sentence scope, so it is useful to decompose the state of a document into the state of its sentences, and we define the overall state S as a sequence of sentence states:

$$S = S_1 S_2 \dots S_N, \tag{1}$$

where N is the number of sentences. This implies that we constrain the decoder to emit exactly one output sentence per input sentence.

Let i be the number of a sentence and m_i the number of input tokens of this sentence, p and q (with $1 \leq p \leq q \leq m_i$) be positions in the input sentence and $[p;q]$ denote the set of positions from p up to and including q. We say that $[p;q]$ precedes $[p';q']$, or $[p;q] \prec [p';q']$, if $q < p'$. Let $\Phi_i([p;q])$ be the set of translations for the source phrase covering positions $[p;q]$ in the input sentence i as given by the phrase table. We call $A = \langle [p;q], \phi \rangle$ an *anchored phrase pair* with coverage $C(A) = [p;q]$ if $\phi \in \Phi_i([p;q])$ is a target phrase translating the source words at positions $[p;q]$. Then a sequence of n_i anchored phrase pairs

$$S_i = A_1 A_2 \dots A_{n_i} \tag{2}$$

is a valid sentence state for sentence i if the following two conditions hold:

1. The coverage sets $C(A_j)$ for j in $1, \dots, n_i$ are mutually disjoint, and
2. the anchored phrase pairs jointly cover the complete input sentence, or

$$\bigcup_{j=1}^{n_i} C(A_j) = [1; m_i]. \tag{3}$$

Let $f(S)$ be a scoring function mapping a state S to a real number. As usual in SMT, it is assumed that the scoring function can be decomposed into a linear combination of K feature functions $h_k(S)$, each with a constant weight λ_k, so

$$f(S) = \sum_{k=1}^{K} \lambda_k h_k(S). \tag{4}$$

The problem addressed by the decoder is the search for the state $\hat{S}$ with maximal score, such that

$$\hat{S} = \arg\max_S f(S). \tag{5}$$

The feature functions implemented in our baseline system are identical to the ones found in the popular Moses SMT system (Koehn et al., 2007). In particular, our decoder has the following feature functions:

1. phrase translation scores provided by the phrase table including forward and backward conditional probabilities, lexical weights and a phrase penalty (Koehn et al., 2003),

2. n-gram language model scores implemented with the KenLM toolkit (Heafield, 2011),

3. a word penalty score,

4. a distortion model with geometric decay (Koehn et al., 2003), and

5. a feature indicating the number of times a given distortion limit is exceeded in the current state.

In our experiments, the last feature is used with a fixed weight of negative infinity in order to limit the gaps between the coverage sets of adjacent anchored phrase pairs to a maximum value. In DP search, the distortion limit is usually enforced directly by the search algorithm and is not added as a feature. In our decoder, however, this restriction is not required to limit complexity, so we decided to add it among the scoring models.

2.2 Decoding Algorithm

The decoding algorithm we use (algorithm 1) is very simple. It starts with a given initial document state. In the main loop, which extends from line 3 to line 12, it generates a successor state S' for the current state S by calling the function `Neighbour`, which non-deterministically applies one of the operations described in section 3 of this paper to S. The score of the new state is compared to that of the previous one. If it meets a given acceptance criterion, S' becomes the current state, else search continues from the previous state S. For the experiments in this paper, we use the *hill climbing acceptance criterion*, which simply accepts a new state if its score is higher than that of the current state. Other acceptance criteria are possible and could be used to endow the search algorithm with stochastic behaviour. The main loop is repeated until a maximum number of steps (*step limit*) is reached or until a maximum number of moves are rejected in a row (*rejection limit*).

Algorithm 1 Decoding algorithm

Input: an initial document state S;
search parameters *maxsteps* and *maxrejected*
Output: a modified document state

```
1:  nsteps ← 0
2:  nrejected ← 0
3:  while nsteps < maxsteps and
    nrejected < maxrejected do
4:    S' ← Neighbour(S)
5:    if Accept(f(S'), f(S)) then
6:      S ← S'
7:      nrejected ← 0
8:    else
9:      nrejected ← nrejected + 1
10:   end if
11:   nsteps ← nsteps + 1
12: end while
13: return S
```

A notable difference between this algorithm and other hill climbing algorithms that have been used for SMT decoding (Germann et al., 2004; Langlais et al., 2007) is its non-determinism. Previous work for sentence-level decoding employed a *steepest ascent* strategy which amounts to enumerating the complete neighbourhood of the current state as defined by the state operations and selecting the next state to be the best state found in the neighbourhood of the current one. Enumerating all neighbours of a given state, costly as it is, has the advantage that it makes it easy to prove local optimality of a state by recognising that all possible successor states have lower scores. It can be rather inefficient, since at every step only one modification will be adopted; many of the modifications that are discarded will very likely be generated anew in the next iteration.

As we extend the decoder to the document level, the size of the neighbourhood that would have to be explored in this way increases considerably. Moreover, the inefficiency of the steepest ascent approach potentially increases as well. Very likely, a promising move in one sentence will remain promising after a modification has been applied to another sen-

tence, even though this is not guaranteed to be true in the presence of cross-sentence models. We therefore adopt a *first-choice hill climbing* strategy that non-deterministically generates successor states and accepts the first one that meets the acceptance criterion. This frees us from the necessity of generating the full set of successors for each state. On the downside, if the full successor set is not known, it is no longer possible to prove local optimality of a state, so we are forced to use a different condition for halting the search. We use a combination of two limits: The step limit is a hard limit on the resources the user is willing to expend on the search problem. The value of the rejection limit determines how much of the neighbourhood is searched for better successors before a state is accepted as a solution; it is related to the probability that a state returned as a solution is in fact locally optimal.

To simplify notations in the description of the individual state operations, we write

$$S_i \longrightarrow S'_i \tag{6}$$

to signify that a state operation, when presented with a document state as in equation 1 and acting on sentence i, returns a new document state of

$$S' = S_1 \ldots S_{i-1}\, S'_i\, S_{i+1} \ldots S_N. \tag{7}$$

Similarly,

$$S_i : A_j \ldots A_{j+h-1} \longrightarrow A'_1 \ldots A'_{h'} \tag{8}$$

is equivalent to

$$S_i \longrightarrow A_1 \ldots A_{j-1}\, A'_1 \ldots A'_{h'}\, A_{j+h} \ldots A_{n_i} \tag{9}$$

and indicates that the operation returns a state in which a sequence of h consecutive anchored phrase pairs has been replaced by another sequence of h' anchored phrase pairs.

2.3 Efficiency Considerations

When implementing the feature functions for the decoder, we have to exercise some care to avoid recomputing scores for the whole document at every iteration. To achieve this, the scores are computed completely only once, at the beginning of the decoding run. In subsequent iterations, scoring functions are presented with the scores of the previous iteration and a list of modifications produced by the state operation, a set of tuples $\langle i, r, s, A'_1 \ldots A'_{h'} \rangle$, each indicating that the document should be modified as described by

$$S_i : A_r \ldots A_s \longrightarrow A'_1 \ldots A'_{h'}. \tag{10}$$

If a feature function is decomposable in some way, as all the standard features developed under the constraints of DP search are, it can then update the state simply by subtracting and adding score components pertaining to the modified parts of the document. Feature functions have the possibility to store their own state information along with the document state to make sure the required information is available. Thus, the framework makes it possible to exploit decomposability for efficient scoring without imposing any particular decomposition on the features as beam search does.

To make scoring even more efficient, scores are computed in two passes: First, every feature function is asked to provide an upper bound on the score that will be obtained for the new state. In some cases, it is possible to calculate reasonable upper bounds much more efficiently than computing the exact feature value. If the upper bound fails to meet the acceptance criterion, the new state is discarded right away; if not, the full score is computed and the acceptance criterion is tested again.

Among the basic SMT models, this two-pass strategy is only used for the n-gram LM, which requires fairly expensive parameter lookups for scoring. The scores of all the other baseline models are fully computed during the first scoring pass. The n-gram model is more complex. In its state information, it keeps track of the LM score and LM library state for each word. The first scoring pass then identifies the words whose LM scores are affected by the current search step. This includes the words changed by the search operation as well as the words whose LM history is modified. The range of the history dependencies can be determined precisely by considering the "valid state length" information provided by the KenLM library. In the first pass, the LM scores of the affected words are subtracted from the total score. The model only looks up the new LM scores for the affected words and updates the total score if the new search state passes the first acceptance check. This two-pass scoring approach allows us

to avoid LM lookups altogether for states that will be rejected anyhow because of low scores from the other models, e. g. because the distortion limit is violated.

Model score updates become more complex and slower as the number of dependencies of a model increases. While our decoding algorithm does not impose any formal restrictions on the number or type of dependencies that can be handled, there will be practical limits beyond which decoding becomes unacceptably slow or the scoring code becomes very difficult to maintain. These limits are however fairly independent of the types of dependencies handled by a model, which permits the exploration of more varied model types than those handled by DP search.

2.4 State Initialisation

Before the hill climbing decoding algorithm can be run, an initial state must be generated. The closer the initial state is to an optimum, the less work remains to be done for the algorithm. If the algorithm is to be self-contained, initialisation must be relatively uninformed and can only rely on some general prior assumptions about what might be a good initial guess. On the other hand, if optimal results are sought after, it pays off to invest some effort into a good starting point. One way to do this is to run DP search first.

For uninformed initialisation, we chose to implement a very simple procedure based only on the observation that, at least for language pairs involving the major European languages, it is usually a good guess to keep the word order of the output very similar to that of the input. We therefore create the initial state by selecting, for each sentence in the document, a sequence of anchored phrase pairs covering the input sentence in monotonic order, that is, such that for all pairs of adjacent anchored phrase pairs A_j and A_{j+1}, we have that $C(A_j) \prec C(A_{j+1})$.

For initialisation with DP search, we first run the Moses decoder (Koehn et al., 2007) with default search parameters and the same models as those used by our decoder. Then we extract the best output hypothesis from the search graph of the decoder and map it into a sequence of anchored phrase pairs in the obvious way. When the document-level decoder is used with models that are incompatible with beam search, Moses can be run with a subset of the models in order to find an approximation of the solution which is then refined with the complete feature set.

3 State Operations

Given a document state S, the decoder uses a neighbourhood function `Neighbour` to simulate a move in the state space. The neighbourhood function nondeterministically selects a type of state operation and a location in the document to apply it to and returns the resulting new state. We use a set of three operations that has the property that every possible document state can be reached from every other state in a sequence of moves.

Designing operations for state transitions in local search for phrase-based SMT is a problem that has been addressed in the literature (Langlais et al., 2007; Arun et al., 2010). Our decoder's first-choice hill climbing strategy never enumerates the full neighbourhood of a state. We therefore place less emphasis than previous work on defining a compact neighbourhood, but allow the decoder to make quite extensive changes to a state in a single step with a certain probability. Otherwise our operations are similar to those used by Arun et al. (2010).

All of the operations described in this paper make changes to a single sentence only. Each time it is called, the `Neighbour` function selects a sentence in the document with a probability proportional to the number of input tokens in each sentence to ensure a fair distribution of the decoder's attention over the words in the document regardless of varying sentence lengths.

3.1 Changing Phrase Translations

The `change-phrase-translation` operation replaces the translation of a single phrase with a random translation with the same coverage taken from the phrase table. Formally, the operation selects an anchored phrase pair A_j by drawing uniformly from the elements of S_i and then draws a new translation ϕ' uniformly from the set $\Phi_i(C(A_j))$. The new state is given by

$$S_i : A_j \longrightarrow \langle C(A_j), \phi' \rangle. \tag{11}$$

3.2 Changing Word Order

The `swap-phrases` operation affects the output word order without changing the phrase translations.

It exchanges two anchored phrase pairs A_j and A_{j+h}, resulting in an output state of

$$S_i : A_j \ldots A_{j+h} \longrightarrow A_{j+h}\, A_{j+1} \ldots A_{j+h-1}\, A_j. \quad (12)$$

The start location j is drawn uniformly from the eligible sentence positions; the swap range h comes from a geometric distribution with configurable decay. Other word-order changes such as a one-way move operation that does not require another movement in exchange or more advanced permutations can easily be defined.

3.3 Resegmentation

The most complex operation is `resegment`, which allows the decoder to modify the segmentation of the source phrase. It takes a number of anchored phrase pairs that form a contiguous block both in the input and in the output and replaces them with a new set of phrase pairs covering the same span of the input sentence. Formally,

$$S_i : A_j \ldots A_{j+h-1} \longrightarrow A'_1 \ldots A'_{h'} \quad (13)$$

such that

$$\bigcup_{j'=j}^{j+h-1} C(A_{j'}) = \bigcup_{j'=1}^{h'} C(A'_{j'}) = [p;q] \quad (14)$$

for some p and q, where, for $j' = 1, \ldots, h'$, we have that $A'_{j'} = \langle [p_{j'}; q_{j'}], \phi_{j'} \rangle$, all $[p_{j'}; q_{j'}]$ are mutually disjoint and each $\phi_{j'}$ is randomly drawn from $\Phi_i([p_{j'}; q_{j'}])$.

Regardless of the ordering of $A_j \ldots A_{j+h-1}$, the `resegment` operation always generates a sequence of anchored phrase pairs in linear order, such that $C(A'_{j'}) \prec C(A'_{j'+1})$ for $j' = 1, \ldots, h' - 1$.

As for the other operations, j is generated uniformly and h is drawn from a geometric distribution with a decay parameter. The new segmentation is generated by extending the sequence of anchored phrase pairs with random elements starting at the next free position, proceeding from left to right until the whole range $[p;q]$ is covered.

4 Experimental Results

In this section, we present the results of a series of experiments with our document decoder. The goal of our experiments is to demonstrate the behaviour of the decoder and characterise its response to changes in the fundamental search parameters.

The SMT models for our experiments were created with a subset of the training data for the English-French shared task at the WMT 2011 workshop (Callison-Burch et al., 2011). The phrase table was trained on Europarl, news-commentary and UN data. To reduce the training data to a manageable size, singleton phrase pairs were removed before the phrase scoring step. Significance-based filtering (Johnson et al., 2007) was applied to the resulting phrase table. The language model was a 5-gram model with Kneser-Ney smoothing trained on the monolingual News corpus with IRSTLM (Federico et al., 2008). Feature weights were trained with Minimum Error-Rate Training (MERT) (Och, 2003) on the news-test2008 development set using the DP beam search decoder and the MERT implementation of the Moses toolkit (Koehn et al., 2007). Experimental results are reported for the newstest2009 test set, a corpus of 111 newswire documents totalling 2,525 sentences or 65,595 English input tokens.

4.1 Stability

An important difference between our decoder and the classical DP decoder as well as previous work in SMT decoding with local search is that our decoder is inherently non-deterministic. This implies that repeated runs of the decoder with the same search parameters, input and models will not, in general, find the same local maximum of the score space. The first empirical question we ask is therefore how different the results are under repeated runs. The results in this and the next section were obtained with random state initialisation, i. e. without running the DP beam search decoder.

Figure 1 shows the results of 7 decoder runs with the models described above, translating the newstest2009 test set, with a step limit of 2^{27} and a rejection limit of 100,000. The x-axis of both plots shows the number of decoding steps on a logarithmic scale, so the number of steps is doubled between two adjacent points on the same curve. In the left plot, the y-axis indicates the model score optimised by the decoder summed over all 2525 sentences of the document. In the right plot, the case-sensitive BLEU score (Papineni et al., 2002) of the current decoder

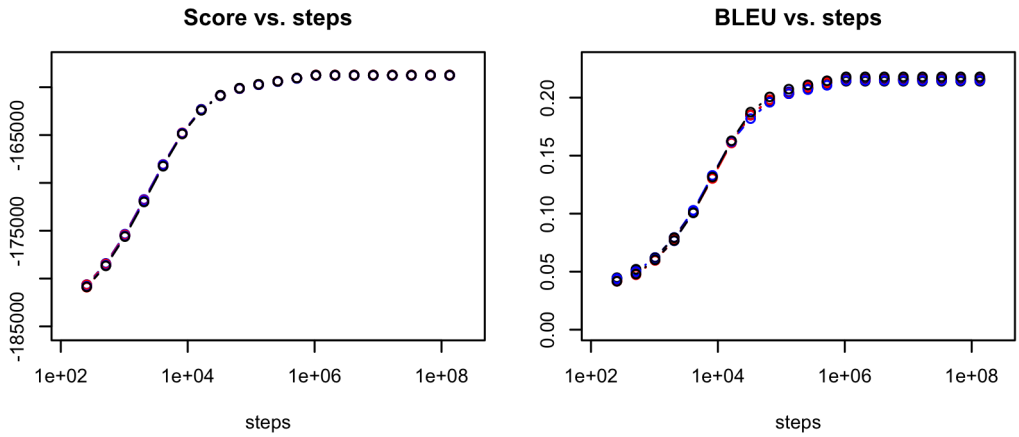

Figure 1: Score stability in repeated decoder runs

state against a reference translation is displayed.

We note, as expected, that the decoder achieves a considerable improvement of the initial state with diminishing returns as decoding continues. Between 2^8 and 2^{14} steps, the score increases at a roughly logarithmic pace, then the curve flattens out, which is partly due to the fact that decoding for some documents effectively stopped when the maximum number of rejections was reached. The BLEU score curve shows a similar increase, from an initial score below 5 % to a maximum of around 21.5 %. This is below the score of 22.45 % achieved by the beam search decoder with the same models, which is not surprising considering that our decoder approximates a more difficult search problem, from which a number of strong independence assumptions have been lifted, without, at the moment, having any stronger models at its disposal to exploit this additional freedom for better translation.

In terms of stability, there are no dramatic differences between the decoder runs. Indeed, the small differences that exist are hardly discernible in the plots. The model scores at the end of the decoding run range between -158767.9 and -158716.9, a relative difference of only about 0.03 %. Final BLEU scores range from 21.41 % to 21.63 %, an interval that is not negligible, but comparable to the variance observed when, e. g., feature weights from repeated MERT runs are used with one and the same SMT system. Note that these results were obtained with random state initialisation. With DP initialisation, score differences between repeated runs rarely exceed 0.02 absolute BLEU percentage points.

Overall, we conclude that the decoding results of our algorithm are reasonably stable despite the non-determinism inherent in the procedure. In our subsequent experiments, the evaluation scores reported are calculated as the mean of three runs for each experiment.

4.2 Search Algorithm Parameters

The hill climbing algorithm we use has two parameters which govern the trade-off between decoding time and the accuracy with which a local maximum is identified: The *step limit* stops the search process after a certain number of steps regardless of the search progress made or lack thereof. The *rejection limit* stops the search after a certain number of unsuccessful attempts to make a step, when continued search does not seem to be promising. In most of our experiments, we used a step limit of $2^{27} \approx 1.3 \cdot 10^8$ and a rejection limit of 10^5. In practice, decoding terminates by reaching the rejection limit for the vast majority of documents. We therefore examined the effect of different rejection limits on the learning curves. The results are shown in figure 2.

The results show that continued search does pay off to a certain extent. Indeed, the curve for rejection limit 10^7 seems to indicate that the model score increases roughly logarithmically, albeit to a higher base, even after the curve has started to flatten out at 2^{14} steps. At a certain point, however, the probability of finding a good successor state drops rather sharply by about two orders of magnitude, as

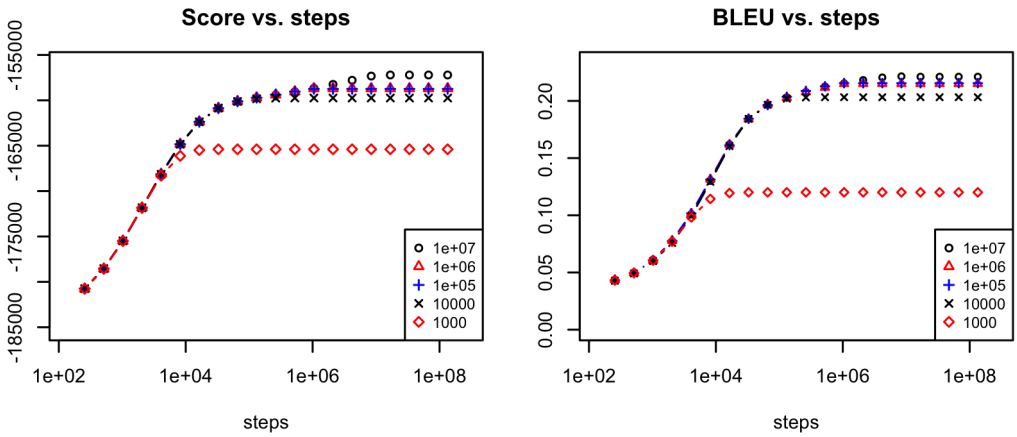

Figure 2: Search performance at different rejection limits

evidenced by the fact that a rejection limit of 10^6 does not give a large improvement over one of 10^5, while one of 10^7 does. The continued model score improvement also results in an increase in BLEU scores, and with a BLEU score of 22.1 % the system with rejection limit 10^7 is fairly close to the score of 22.45 % obtained by DP beam search.

Obviously, more exact search comes at a cost, and in this case, it comes at a considerable cost, which is an explosion of the time required to decode the test set from 4 minutes at rejection limit 10^3 to 224 minutes at rejection limit 10^5 and 38 hours 45 minutes at limit 10^7. The DP decoder takes 31 minutes for the same task. We conclude that the rejection limit of 10^5 selected for our experiments, while technically suboptimal, realises a good trade-off between decoding time and accuracy.

4.3 A Semantic Document Language Model

In this section, we present the results of the application of our decoder to an actual SMT model with cross-sentence features. Our model addresses the problem of *lexical cohesion*. In particular, it rewards the use of semantically related words in the translation output by the decoder, where semantic distance is measured with a word space model based on Latent Semantic Analysis (LSA). LSA has been applied to semantic language modelling in previous research with some success (Coccaro and Jurafsky, 1998; Bellegarda, 2000; Wandmacher and Antoine, 2007). In SMT, it has mostly been used for domain adaptation (Kim and Khudanpur, 2004; Tam et al., 2007), or to measure sentence similarities (Banchs and Costa-jussà, 2011).

The model we use is inspired by Bellegarda (2000). It is a Markov model, similar to a standard n-gram model, and assigns to each content word a score given a history of n preceding content words, where $n = 30$ below. Scoring relies on a 30-dimensional LSA word vector space trained with the S-Space software (Jurgens and Stevens, 2010). The score is defined based on the cosine similarity between the word vector of the predicted word and the mean word vector of the words in the history, which is converted to a probability by histogram lookup as suggested by Bellegarda (2000). The model is structurally different from a regular n-gram model in that word vector n-grams are defined over content words occurring in the word vector model only and can cross sentence boundaries. Stop words, identified by an extensive stop word list and amounting to around 60 % of the tokens, are scored by a different mechanism based on their relative frequency (undiscounted unigram probability) in the training corpus. In sum, the score produced by the semantic document LM has the following form:

$$h(w|h) = \begin{cases} p_{\text{unigr}}(w) & \text{if } w \text{ is a stop word, else} \\ \alpha p_{\cos}(w|h) & \text{if } w \text{ is known, else} \\ \varepsilon & \text{if } w \text{ is unknown,} \end{cases} \quad (15)$$

where α is the proportion of content words in the training corpus and ε is a small fixed probability. It is integrated into the decoder as an extra feature function. Since we lack an automatic method for

training the feature weights of document-wide features, its weight was selected by grid search over a number of values, comparing translation performance for the newstest2009 test set.

In these experiments, we used DP beam search to initialise the state of our local search decoder. Three results are presented (table 1): The first table row shows the baseline performance using DP beam search with standard sentence-local features only. The scores in the second row were obtained by running the hill climbing decoder with DP initialisation, but without adding any models. A marginal increase in scores for all three test sets demonstrates that the hill climbing decoder manages to fix some of the search errors made by the DP search. The last row contains the scores obtained by adding in the semantic language model. Scores are presented for three publicly available test sets from recent WMT Machine Translation shared tasks, of which one (newstest2009) was used to monitor progress during development and select the final model.

Adding the semantic language model results in a small increase in NIST scores (Doddington, 2002) for all three test sets as well as a small BLEU score gain (Papineni et al., 2002) for two out of three corpora. We note that the NIST score turned out to react more sensitively to improvements due to the semantic LM in all our experiments, which is reasonable because the model specifically targets content words, which benefit from the information weighting done by the NIST score. While the results we present do not constitute compelling evidence in favour of our semantic LM in its current form, they do suggest that this model could be improved to realise higher gains from cross-sentence semantic information. They support our claim that cross-sentence models should be examined more closely and that existing methods should be adapted to deal with them, a problem addressed by our main contribution, the local search document decoder.

5 Related Work

Even though DP beam search (Koehn et al., 2003) has been the dominant approach to SMT decoding in recent years, methods based on local search have been explored at various times. For word-based SMT, greedy hill-climbing techniques were advocated as a faster replacement for beam search (Germann et al., 2001; Germann, 2003; Germann et al., 2004), and a problem formulation specifically targeting word reordering with an efficient word reordering algorithm has been proposed (Eisner and Tromble, 2006).

A local search decoder has been advanced as a faster alternative to beam search also for phrase-based SMT (Langlais et al., 2007; Langlais et al., 2008). That work anticipates many of the features found in our decoder, including the use of local search to refine an initial hypothesis produced by DP beam search. The possibility of using models that do not fit well into the beam search paradigm is mentioned and illustrated with the example of a reversed n-gram language model, which the authors claim would be difficult to implement in a beam search decoder. Similarly to the work by Germann et al. (2001), their decoder is deterministic and explores the entire neighbourhood of a state in order to identify the most promising step. Our main contribution with respect to the work by Langlais et al. (2007) is the introduction of the possibility of handling document-level models by lifting the assumption of sentence independence. As a consequence, enumerating the entire neighbourhood becomes too expensive, which is why we resort to a "first-choice" strategy that non-deterministically generates states and accepts the first one encountered that meets the acceptance criterion.

More recently, Gibbs sampling was proposed as a way to generate samples from the posterior distribution of a phrase-based SMT decoder (Arun et al., 2009; Arun et al., 2010), a process that resembles local search in its use of a set of state-modifying operators to generate a sequence of decoder states. Where local search seeks for the best state attainable from a given initial state, Gibbs sampling produces a representative sample from the posterior. Like all work on SMT decoding that we know of, the Gibbs sampler presented by Arun et al. (2010) assumes independence of sentences and considers the complete neighbourhood of each state before taking a sample.

6 Conclusion

In the last twenty years of SMT research, there has been a strong assumption that sentences in a text

	newstest2009		newstest2010		newstest2011	
	BLEU	NIST	BLEU	NIST	BLEU	NIST
DP search only	22.56	6.513	27.27	7.034	24.94	7.170
DP + hill climbing	22.60	6.518	27.33	7.046	24.97	7.169
with semantic LM	22.71	6.549	27.53	7.087	24.90	7.199

Table 1: Experimental results with a cross-sentence semantic language model

are independent of one another, and discourse context has been largely neglected. Several factors have contributed to this. Developing good discourse-level models is difficult, and considering the modest translation quality that has long been achieved by SMT, there have been more pressing problems to solve and lower hanging fruit to pick. However, we argue that the popular DP beam search algorithm, which delivers excellent decoding performance, but imposes a particular kind of local dependency structure on the feature models, has also had its share in driving researchers away from discourse-level problems.

In this paper, we have presented a decoding procedure for phrase-based SMT that makes it possible to define feature models with cross-sentence dependencies. Our algorithm can be combined with DP beam search to leverage the quality of the traditional approach with increased flexibility for models at the discourse level. We have presented preliminary results on a cross-sentence semantic language model addressing the problem of lexical cohesion to demonstrate that this kind of models is worth exploring further. Besides lexical cohesion, cross-sentence models are relevant for other linguistic phenomena such as pronominal anaphora or verb tense selection. We believe that SMT research has reached a point of maturity where discourse phenomena should not be ignored any longer, and we consider our decoder to be a step towards this goal.

References

Abhishek Arun, Chris Dyer, Barry Haddow, Phil Blunsom, Adam Lopez, and Philipp Koehn. 2009. Monte carlo inference and maximization for phrase-based translation. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 102–110, Boulder, Colorado, June. Association for Computational Linguistics.

Abhishek Arun, Barry Haddow, Philipp Koehn, Adam Lopez, Chris Dyer, and Phil Blunsom. 2010. Monte Carlo techniques for phrase-based translation. *Machine translation*, 24(2):103–121.

Rafael E. Banchs and Marta R. Costa-jussà. 2011. A semantic feature for Statistical Machine Translation. In *Proceedings of Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 126–134, Portland, Oregon, USA, June. Association for Computational Linguistics.

Jerome R. Bellegarda. 2000. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, 88(8):1279–1296.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan. 2011. Findings of the 2011 Workshop on Statistical Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, Scotland, July. Association for Computational Linguistics.

Noah Coccaro and Daniel Jurafsky. 1998. Towards better integration of semantic predictors in statistical language modeling. In *Proceedings of the 5th International Conference on Spoken Language Processing*, Sydney.

George Doddington. 2002. Automatic evaluation of machine translation quality using *n*-gram co-occurrence statistics. In *Proceedings of the second International conference on Human Language Technology Research*, pages 138–145, San Diego.

Jason Eisner and Roy W. Tromble. 2006. Local search with very large-scale neighborhoods for optimal permutations in machine translation. In *Proceedings of the HLT-NAACL Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pages 57–75.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. IRSTLM: an open source toolkit for handling large scale language models. In *Interspeech 2008*, pages 1618–1621. ISCA.

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

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2004. Fast and optimal decoding for machine translation. *Artificial Intelligence*, 154(1–2):127–143.

Ulrich Germann. 2003. Greedy decoding for Statistical Machine Translation in almost linear time. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based document-level Statistical Machine Translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 909–919, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Christian Hardmeier and Marcello Federico. 2010. Modelling Pronominal Anaphora in Statistical Machine Translation. In *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*, pages 283–289.

Basil Hatim and Ian Mason. 1990. *Discourse and the Translator*. Language in Social Life Series. Longman, London.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July. Association for Computational Linguistics.

Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. 2007. Improving translation quality by discarding most of the phrasetable. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 967–975, Prague, Czech Republic, June. Association for Computational Linguistics.

David Jurgens and Keith Stevens. 2010. The S-Space package: An open source package for word space models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 30–35, Uppsala, Sweden, July. Association for Computational Linguistics.

Woosung Kim and Sanjeev Khudanpur. 2004. Cross-lingual latent semantic analysis for language modeling. In *IEEE international conference on acoustics, speech, and signal processing (ICASSP)*, volume 1, pages 257–260, Montréal.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 conference of the North American chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Edmonton.

Philipp Koehn, Hieu Hoang, Alexandra Birch, et al. 2007. Moses: open source toolkit for Statistical Machine Translation. In *Annual meeting of the Association for Computational Linguistics: Demonstration session*, pages 177–180, Prague.

Philippe Langlais, Alexandre Patry, and Fabrizio Gotti. 2007. A greedy decoder for phrase-based statistical machine translation. In *TMI-2007: Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 104–113, Skövde.

Philippe Langlais, Alexandre Patry, and Fabrizio Gotti. 2008. Recherche locale pour la traduction statistique par segments. In *TALN 2008*, pages 119–128, Avignon, France, June. ATALA.

Ronan Le Nagard and Philipp Koehn. 2010. Aiding pronoun translation with co-reference resolution. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 252–261, Uppsala, Sweden, July. Association for Computational Linguistics.

Franz Josef Och, Nicola Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for Statistical Machine Translation. In *Proceedings of the Data-Driven Machine Translation Workshop, 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 55–62, Toulouse.

Franz Josef Och. 2003. Minimum error rate training in Statistical Machine Translation. In *Proceedings of the 41st annual meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo (Japan).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of Machine Translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia. ACL.

Yik-Cheung Tam, Ian Lane, and Tanja Schultz. 2007. Bilingual LSA-based adaptation for Statistical Machine Translation. *Machine Translation*, 21(4):187–207.

Jörg Tiedemann. 2010. To cache or not to cache? Experiments with adaptive models in Statistical Machine Translation. In *Proceedings of the ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR*, pages 189–194, Uppsala, Sweden. Association for Computational Linguistics.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a Dynamic Programming beam search algorithm for Statistical Machine Translation. *Computational linguistics*, 29(1):97–133.

Christoph Tillmann, Stephan Vogel, Hermann Ney, and Alex Zubiaga. 1997. A DP-based search using monotone alignments in Statistical Translation. In *Proceedings of the 35th Annual Meeting of the Association for*

Computational Linguistics, pages 289–296, Madrid, Spain, July. Association for Computational Linguistics.

Tonio Wandmacher and Jean-Yves Antoine. 2007. Methods to integrate a language model with semantic information for a word prediction component. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 506–513, Prague, Czech Republic, June. Association for Computational Linguistics.