Generalized Stack Decoding Algorithms for Statistical Machine Translation[*]

Daniel Ortiz Martínez
Inst. Tecnológico de Informática
Univ. Politécnica de Valencia
46071 Valencia, Spain
dortiz@iti.upv.es

Ismael García Varea
Dpto. de Informatica
Univ. de Castilla-La Mancha
02071 Albacete, Spain
ivarea@info-ab.uclm.es

Francisco Casacuberta Nolla
Dpto. de Sist Inf. y Comp.
Univ. Politéc. de Valencia
46071 Valencia, Spain
fcn@dsic.upv.es

Abstract

In this paper we propose a generalization of the Stack-based decoding paradigm for Statistical Machine Translation. The well known single and multi-stack decoding algorithms defined in the literature have been integrated within a new formalism which also defines a new family of stack-based decoders. These decoders allows a tradeoff to be made between the advantages of using only one or multiple stacks. The key point of the new formalism consists in parameterizeing the number of stacks to be used during the decoding process, and providing an efficient method to decide in which stack each partial hypothesis generated is to be inserted-during the search process. Experimental results are also reported for a search algorithm for phrase-based statistical translation models.

1 Introduction

The translation process can be formulated from a statistical point of view as follows: A source language string $f_1^J = f_1 \ldots f_J$ is to be translated into a target language string $e_1^I = e_1 \ldots e_I$. Every target string is regarded as a possible translation for the source language string with maximum a-posteriori probability $Pr(e_1^I|f_1^J)$. According to Bayes' theorem, the target string $\hat{e}_1^I$ that maximizes[1] the product of both the target language model $Pr(e_1^I)$ and the string translation model $Pr(f_1^J|e_1^I)$ must be chosen. The equation that models this process is:

$$\hat{e}_1^I = \arg\max_{e_1^I}\{Pr(e_1^I) \cdot Pr(f_1^J|e_1^I)\} \qquad (1)$$

The search/decoding problem in SMT consists in solving the maximization problem stated in Eq. (1). In the literature, we can find different techniques to deal with this problem, ranging from heuristic and fast (as greedy decoders) to optimal and very slow decoding algorithms (Germann et al., 2001). Also, under certain circumstances, stack-based decoders can obtain optimal solutions.

Many works (Berger et al., 1996; Wang and Waibel, 1998; Germann et al., 2001; Och et al., 2001; Ortíz et al., 2003) have adopted different types of stack-based algorithms to solve the global search optimization problem for statistical machine translation. All these works follow two main different approaches according to the number of stacks used in the design and implementation of the search algorithm (the stacks are used to store partial hypotheses, sorted according to their partial score/probability, during the search process) :

- On the one hand, in (Wang and Waibel, 1998; Och et al., 2001) a single stack is used. In that case, in order to make the search feasible, the pruning of the number of partial hypotheses stored in the stack is needed. This causes many search errors due to the fact that hypotheses covering a different number of source (translated) words compete in the same conditions. Therefore, the greater number of covered words the higher possibility to be pruned.

- On the other hand (Berger et al., 1996; Germann et al., 2001) make use of multiple stacks

[*]This work has been partially supported by the Spanish project TIC2003-08681-C02-02, the *Agencia Valenciana de Ciencia y Tecnología* under contract GRUPOS03/031, the *Generalitat Valenciana*, and the project HERMES (Vicerrectorado de Investigación - UCLM-05/06)

[1]Note that the expression should also be maximized by I; however, for the sake of simplicity we suppose that it is known.

(one for each set of source covered/translated words in the partial hypothesis) in order to solve the disadvantages of the single-stack approach. By contrast, the problem of finding the best hypothesis to be expanded introduces an exponential term in the computational complexity of the algorithm.

In (Ortíz et al., 2003) the authors present an empirical comparison (about efficiency and translation quality) of the two approaches paying special attention to the advantages and disadvantages of the two approaches.

In this paper we present a new formalism consisting of a generalization of the classical stack-based decoding paradigm for SMT. This new formalism defines a new family of stack-based decoders, which also integrates the well known stack-based decoding algorithms proposed so far within the framework of SMT, that is single and multi-stack decoders.

The rest of the paper is organized as follows: in section 2 the phrase-based approach to SMT is depicted; in section 3 the main features of classical stack-based decoders are presented; in section 4 the new formalism is presented and in section 5 experimental results are shown; finally some conclusions are drawn in section 6.

2 Phrase Based Statistical Machine Translation

Different *translation models* (TMs) have been proposed depending on how the relation between the source and the target languages is structured; that is, the way a target sentence is generated from a source sentence. This relation is summarized using the concept of *alignment*; that is, how the constituents (typically words or group-of-words) of a pair of sentences are aligned to each other. The most widely used single-word-based *statistical alignment models* (SAMs) have been proposed in (Brown et al., 1993; Ney et al., 2000). On the other hand, models that deal with structures or phrases instead of single words have also been proposed: the syntax translation models are described in (Yamada and Knight, 2001) , alignment templates are used in (Och, 2002), and the alignment template approach is re-framed into the so-called *phrase based translation* (PBT) in (Marcu and Wong, 2002; Zens et al., 2002; Koehn et al., 2003; Tomás and Casacuberta, 2003).

For the translation model ($Pr(f_1^J|e_1^I)$) in Eq. (1), PBT can be explained from a generative point of view as follows (Zens et al., 2002):

1. The target sentence e_1^I is segmented into K phrases ($\tilde{e}_1^K$).
2. Each target phrase $\tilde{e}_k$ is translated into a source phrase $\tilde{f}$.
3. Finally, the source phrases are reordered in order to compose the source sentence $\tilde{f}_1^K = f_1^J$.

In PBT, it is assumed that the relations between the words of the source and target sentences can be explained by means of the hidden variable $\tilde{a}_1^K$, which contains all the decisions made during the generative story.

$$\begin{aligned} Pr(f_1^J|e_1^I) &= \sum_{K,\tilde{a}_1^K} Pr(,\tilde{f}_1^K,\tilde{a}_1^K|\tilde{e}_1^K) \\ &= \sum_{K,\tilde{a}_1^K} Pr(\tilde{a}_1^K|\tilde{e}_1^K) Pr(\tilde{f}_1^K|\tilde{a}_1^K,\tilde{e}_1^K) \end{aligned} \tag{2}$$

Different assumptions can be made from the previous equation. For example, in (Zens et al., 2002) the following model is proposed:

$$p_\theta(f_1^J|e_1^I) = \alpha(e_1^I) \sum_{K,\tilde{a}_1^K} \prod_{k=1}^{K} p(\tilde{f}_k|\tilde{e}_{\tilde{a}_k}) \tag{3}$$

where $\tilde{a}_k$ notes the index of the source phrase $\tilde{e}$ which is aligned with the k-th target phrase $\tilde{f}_k$ and that all possible segmentations have the same probability. In (Tomás and Casacuberta, 2001; Zens et al., 2002), it also is assumed that the alignments must be monotonic. This led us to the following equation:

$$p_\theta(f_1^J|e_1^I) = \alpha(e_1^I) \sum_{K,\tilde{a}_1^K} \prod_{k=1}^{K} p(\tilde{f}_k|\tilde{e}_k) \tag{4}$$

In both cases the model parameters that have to be estimated are the translation probabilities between phrase pairs ($\theta = \{p(\tilde{f}|\tilde{e})\}$), which typically are estimated as follows:

$$p(\tilde{f}|\tilde{e}) = \frac{N(\tilde{f},\tilde{e})}{N(\tilde{e})} \tag{5}$$

where $N(\tilde{f}|\tilde{e})$ is the number of times that $\tilde{f}$ have been seen as a translation of $\tilde{e}$ within the training corpus.

3 Stack-Decoding Algorithms

The stack decoding algorithm, also called A^* algorithm, was first introduced by F. Jelinek in (Jelinek, 1969). The stack decoding algorithm attempts to generate partial solutions, called *hypotheses*, until a complete translation is found[2]; these hypotheses are stored in a stack and ordered by their *score*. Typically, this measure or score is the probability of the product of the translation and the language models introduced above. The A^* decoder follows a sequence of steps for achieving a complete (and possibly optimal) hypothesis:

1. Initialize the stack with an empty hypothesis.
2. Iterate
   (a) Pop h (the best hypothesis) off the stack.
   (b) If h is a complete sentence, output h and terminate.
   (c) Expand h.
   (d) Go to step 2a.

The search is started from a null string and obtains new hypotheses after an expansion process (step 2c) which is executed at each iteration. The expansion process consists of the application of a set of operators over the best hypothesis in the stack, as it is depicted in Figure 1. Thus, the design of stack decoding algorithms involves defining a set of operators to be applied over every hypothesis as well as the way in which they are combined in the expansion process. Both the operators and the expansion algorithm depend on the translation model that we use. For the case of the phrase-based translation models described in the previous section, the operator add is defined, which adds a sequence of words to the target sentence, and aligns it with a sequence of words of the source sentence.

The number of hypotheses to be stored during the search can be huge. In order then to avoid memory overflow problems, the maximum number of hypotheses that a stack may store has to be limited. It is important to note that for a hypothesis, the higher the aligned source words, the worse the score. These hypotheses will be discarded sooner when an A^* search algorithm is used due to the stack length limitation. Because of this, the *multi-stack algorithms* were introduced.

[2]Each hypothesis has associated a coverage vector of length J, which indicates the set of source words already covered/translated so far. In the following we will refer to this simply as coverage.

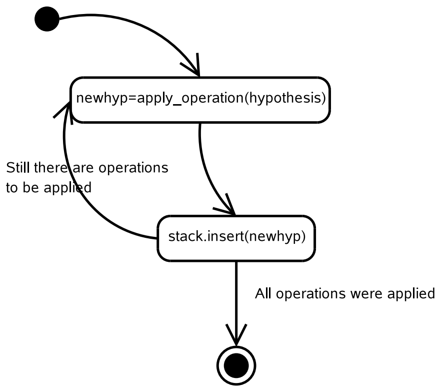
newhyp=apply_operation(hypothesis)
Still there are operations to be applied
stack.insert(newhyp)
All operations were applied

Figure 1: Flow chart associated to the expansion of a hypothesis when using an $A^\star$ algorithm.

Multi-stack algorithms store those hypotheses with different subsets of source aligned words in different stacks. That is to say, given an input sentence f_1^J composed of J words, multi-stack algorithms employes 2^J stacks to translate it. Such an organization improves the pruning of the hypotheses when the stack length limitation is exceeded, since only hypotheses with the same number of covered positions can compete with each other.

All the search steps given for A^* algorithm can also be applied here, except step 2a. This is due to the fact that multiple stacks are used instead of only one. Figure 2 depicts the expansion process that the multi-stack algorithms execute, which is slightly different than the one presented in Figure 1. Multi-stack algorithms have the negative property of spending significant amounts of time in selecting the hypotheses to be expanded, since at each iteration, the best hypothesis in a set of 2^J stacks must be searched for (Ortíz et al., 2003). By contrast, for the A^* algorithm, it is not possible to reduce the length of the stack in the same way as in the multi-stack case without loss of translation quality.

Additionally, certain translation systems, e.g. the Pharaoh decoder (Koehn, 2003) use an alternative

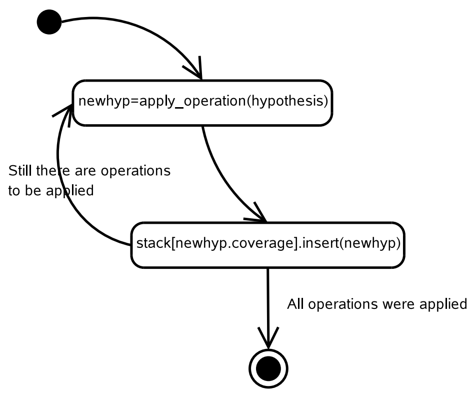
newhyp=apply_operation(hypothesis)
Still there are operations to be applied
stack[newhyp.coverage].insert(newhyp)
All operations were applied

Figure 2: Flow chart associated to the expansion of a hypothesis when using a multi-stack algorithm.

approach which consists in assigning to the same stack, those hypotheses with the same number of source words covered.

4 Generalized Stack-Decoding Algorithms

As was mentioned in the previous section, given a sentence f_1^J to be translated, a single stack decoding algorithm employs only one stack to perform the translation process, while a multi-stack algorithm employs 2^J stacks. We propose a possible way to make a tradeoff between the advantages of both algorithms that introduces a new parameter which will be referred to as the *granularity* of the algorithm. The granularity parameter determines the number of stacks used during the decoding process.

4.1 Selecting the *granularity* of the algorithm

The granularity (G) of a generalized stack algorithm is an integer which takes values between 1 and J, where J is the number of words which compose the sentence to translate.

Given a sentence f_1^J to be translated, a generalized stack algorithm with a granularity parameter equal to g, will have the following features:

- The algorithm will use at most 2^g stacks to perform the translation
- Each stack will contain hypotheses which have 2^{J-g} different coverages of f_1^J
- If the algorithm can store at most $S = s$ hypotheses, then, the maximum size of each stack will be equal to $\frac{s}{2^g}$

4.2 Mapping hypotheses to stacks

Generalized stack-decoding algorithms require a mechanism to decide in which stack each hypothesis is to be inserted. As stated in section 4.1, given an input sentence f_1^J and a generalized stack-decoding algorithm with $G = g$, the decoder will work with 2^g stacks, and each one will contain 2^{J-g} different coverages. Therefore, the above mentioned mechanism can be expressed as a function which will be referred to as the μ function. Given a hypothesis coverage composed of J bits, the μ function return a stack identifier composed of only g bits:

$$\mu : (\{0,1\})^J \longrightarrow (\{0,1\})^g \tag{6}$$

Generalized stack algorithms are strongly inspired by multi-stack algorithms; however, both types of algorithms differ in the way the hypothesis expansion is performed. Figure 3 shows the expansion algorithm of a generalized stack decoder with a granularity parameter equal to g and a function μ which maps hypotheses coverages to stacks.

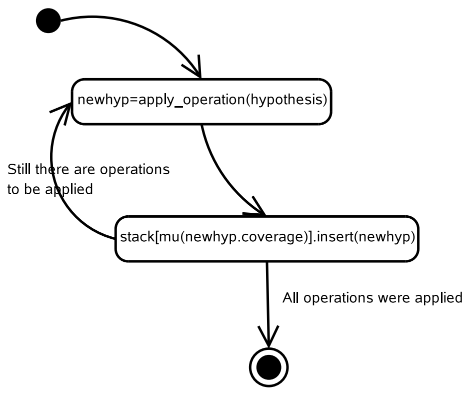
newhyp=apply_operation(hypothesis)
Still there are operations to be applied
stack[mu(newhyp.coverage)].insert(newhyp)
All operations were applied

Figure 3: Flow chart associated to the expansion of a hypothesis when using a generalized-stack algorithm.

The function μ can be defined in many ways, but there are two essential principles which must be taken into account:

- The μ function must be efficiently calculated
- Hypotheses whose coverage have a similar number of bits set to one must be assigned to the same stack. This requirement allows the pruning of the stacks to be improved, since the

hypotheses with a similar number of covered words can compete fairly

A possible way to implement the μ function, namely μ_1, consists in simply shifting the coverage vector $J - g$ positions to the right, and then keeping only the first g bits. Such a proposal is very easy to calculate, however, it has a poor performance according to the second principle explained above.

A better alternative to implement the μ function, namely μ_2, can be formulated as a composition of two functions. A constructive definition of such a implementation is detailed next:

1. Let us suppose that the source sentence is composed by J words, we order the set of J bit numbers as follows: first the numbers which do not have any bit equal to one, next, the numbers which have only one bit equal to one, and so on

2. Given the list of numbers described above, we define a function which associates to each number of the list, the order of the number within this list

3. Given the coverage of a partial hypothesis, x, the stack on which this partial hypothesis is to be inserted is obtained by a two step process: First, we obtain the image of x returned by the function described above. Next, the result is shifted $J - g$ positions to the right, keeping the first g bits

Let β be the function that shifts a bit vector $J - g$ positions to the right, keeping the first g bits; and let α be the function that for each coverage returns its order:

$$\alpha : (\{0,1\})^J \longrightarrow (\{0,1\})^J \qquad (7)$$

Then, μ_2 is expressed as follows:

$$\mu_2(x) = \beta \circ \alpha(x) \qquad (8)$$

Table 1 shows an example of the values which returns the μ_1 and the μ_2 functions when the input sentence has 4 words and the granularity of the decoder is equal to 2. As it can be observed, μ_2 function performs better than μ_1 function according to the second principle described at the beginning of this section.

x	$\mu_1(x)$	$\alpha(x)$	$\mu_2(x)$
0000	00	0000	00
0001	00	0001	00
0010	00	0010	00
0100	01	0011	00
1000	10	0100	01
0011	00	0101	01
0101	01	0110	01
0110	01	0111	01
1001	10	1000	10
1010	10	1001	10
1100	11	1010	10
0111	01	1011	10
1011	10	1100	11
1101	11	1101	11
1110	11	1110	11
1111	11	1111	11

Table 1: Values returned by the μ_1 and μ_2 function defined as a composition of the α and β functions

4.3 Single and Multi Stack Algorithms

The classical single and multi-stack decoding algorithms can be expressed/instantiated as particular cases of the general formalism that have been proposed.

Given the input sentence f_1^J , a generalized stack decoding algorithm with $G = 0$ will have the following features:

- The algorithm works with $2^0 = 1$ stacks.

- Such a stack may store hypotheses with 2^J different coverages. That is to say, all possible coverages.

- The mapping function returns the same stack identifier for each coverage

The previously defined algorithm has the same features as a single stack algorithm.

Let us now consider the features of a generalized stack algorithm with a granularity value of J:

- The algorithm works with 2^J stacks

- Each stack may store hypotheses with only $2^0 = 1$ coverage.

- The mapping function returns a different stack identifier for each coverage

The above mentioned features characterizes the multi-stack algorithms described in the literature.

		EuTrans-I		Xerox	
		Spanish	**English**	**Spanish**	**English**
Training	Sentences	10,000		55,761	
	Words	97,131	99,292	753,607	665,400
	Vocabulary size	686	513	11,051	7,957
	Average sentence leng.	9.7	9.9	13.5	11.9
Test	Sentence	2,996		1,125	
	Words	35,023	35,590	10,106	8,370
	Perplexity (Trigrams)	–	3.62	–	48.3

Table 2: EuTrans-I and Xerox corpus statistics

5 Experiments and Results

In this section, experimental results are presented for two well-known tasks: the EuTrans-I (Amengual et al., 1996), a small size and easy translation task, and the Xerox (Cubel et al., 2004), a medium size and difficult translation task. The main statistics of these corpora are shown in Table 2. The translation results were obtained using a non-monotone generalized stack algorithm. For both tasks, the training of the different phrase models was carried out using the publicly available *Thot* toolkit (Ortiz et al., 2005).

Different translation experiments have been carried out, varying the value of G (ranging from 0 to 8) and the maximum number of hypothesis that the algorithm is allow to store for all used stacks (S) (ranging from 2^8 to 2^{12}). In these experiments the following statistics are computed: the average score (or logProb) that the phrase-based translation model assigns to each hypothesis, the translation quality (by means of WER and Bleu measures), and the average time (in secs.) per sentence[3].

In Figures 4 and 5 two plots are shown: the average time per sentence (left) and the average score (right), for EuTrans and Xerox corpora respectively. As can be seen in both figures, the bigger the value of G the lower the average time per sentence. This is true up to the value of $G = 6$. For higher values of G (keeping fixed the value of S) the average time per sentence increase slightly. This is due to the fact that at this point the algorithm start to spend more time to decide which hypothesis is to be expanded. With respect to the average score similar values are obtained up to the value of $G = 4$. Higher values of G slightly decreases the average score. In this case, as G increases, the number of hypotheses per stack decreases, taking into account that the value of S is fixed, then the "optimal" hypothesis can easily be pruned.

In tables 3 and 4 detailed experiments are shown for a value of $S = 2^{12}$ and different values of G, for EuTrans and Xerox corpora respectively.

G	WER	Bleu	secsXsent	logprob
0	6.6	0.898	2.4	-18.88
1	6.6	0.898	1.9	-18.80
2	6.6	0.897	1.7	-18.81
4	6.6	0.898	1.3	-18.77
6	6.7	0.896	1.1	-18.83
8	6.7	0.896	1.5	-18.87

Table 3: Translation experiments for EuTrans corpus using a generalized stack algorithm with different values of G and a fixed value of $S = 2^{12}$

G	WER	Bleu	secsXsent	logProb
0	32.6	0.658	35.1	-33.92
1	32.8	0.657	20.4	-33.86
2	33.1	0.656	12.8	-33.79
4	32.9	0.657	7.0	-33.70
6	33.7	0.652	6.3	-33.69
8	36.3	0.634	13.7	-34.10

Table 4: Translation experiments for Xerox corpus using a generalized stack algorithm with different values of G and a fixed value of $S = 2^{12}$

According to the experiments presented here we can conclude that:

- The results correlates for the two considered tasks: one small and easy, and other larger and difficult.
- The proposed generalized stack decoding paradigm can be used to make a tradeoff be-

[3]All the experiments have been executed on a PC with a 2.60 Ghz Intel Pentium 4 processor with 2GB of memory. All the times are given in seconds.

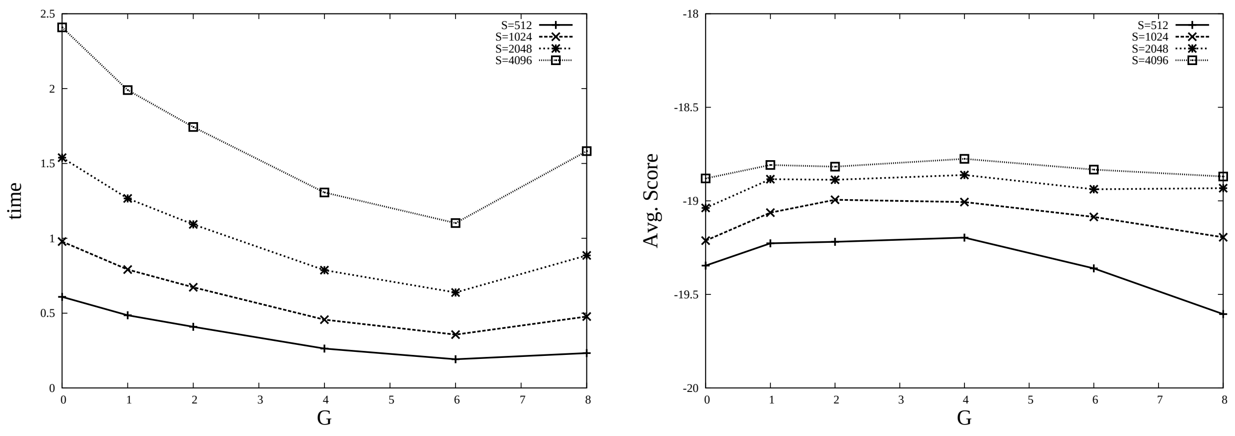

Figure 4: Average time per sentence (in secs.) and average score per sentence. The results are shown for different values of G and S for the EUTRANS corpus.

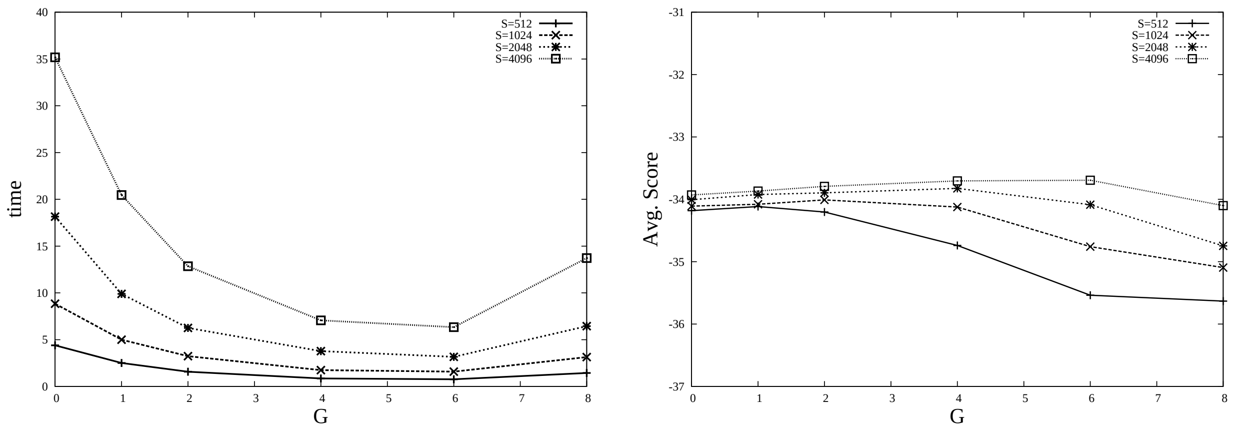

Figure 5: Average time per sentence (in secs.) and average score per sentence. The results are shown for different values of G and S for the XEROX corpus.

tween the advantages of classical single and multi-stack decoding algorithms.

- As we expected, better results (regarding efficiency and accuracy) are obtained when using a value of G between 0 and J.

6 Concluding Remarks

In this paper, a generalization of the stack-decoding paradigm has been proposed. This new formalism includes the well known single and multi-stack decoding algorithms and a new family of stack-based algorithms which have not been described yet in the literature.

Essentially, generalized stack algorithms use a parameterized number of stacks during the decoding process, and try to assign hypotheses to stacks such that there is ”fair competition” within each stack, i.e., brother hypotheses should cover roughly the same number of input words (and the same words) if possible.

The new family of stack-based algorithms allows a tradeoff to be made between the classical single and multi-stack decoding algorithms. For this purpose, they employ a certain number of stacks between 1 (the number of stacks used by a single stack algorithm) and 2^J (the number of stacks used by a multiple stack algorithm to translate a sentence with J words.)

According to the experimental results, it has been proved that an appropriate value of G yields in a stack decoding algorithm that outperforms (in efficiency and acuraccy) the single and multi-stack algorithms proposed so far.

As future work, we plan to extend the experimentation framework presented here to larger and more complex tasks as HANSARDS and EUROPARL corpora.

References

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

Adam L. Berger, Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, John R. Gillett, A. S. Kehler, and R. L. Mercer. 1996. Language translation apparatus and method of using context-based translation models. United States Patent, No. 5510981, April.

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

E. Cubel, J. Civera, J. M. Vilar, A. L. Lagarda, E. Vidal, F. Casacuberta, D. Picó, J. González, and L. Rodríguez. 2004. Finite-state models for computer assisted translation. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI04)*, pages 586–590, Valencia, Spain, August. IOS Press.

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

F. Jelinek. 1969. A fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*, 13:675–685.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the HLT/NAACL*, Edmonton, Canada, May.

Phillip Koehn. 2003. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. User manual and description. Technical report, USC Information Science Institute, December.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the EMNLP Conference*, pages 1408–1414, Philadelphia, USA, July.

Hermann Ney, Sonja Nießen, Franz J. Och, Hassan Sawaf, Christoph Tillmann, and Stephan Vogel. 2000. Algorithms for statistical translation of spoken language. *IEEE Trans. on Speech and Audio Processing*, 8(1):24–36, January.

Franz J. Och, Nicola Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for statistical machine translation. In *Data-Driven Machine Translation Workshop*, pages 55–62, Toulouse, France, July.

Franz Joseph Och. 2002. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. Ph.D. thesis, Computer Science Department, RWTH Aachen, Germany, October.

D. Ortíz, Ismael García-Varea, and Francisco Casacuberta. 2003. An empirical comparison of stack-based decoding algorithms for statistical machine translation. In *New Advance in Computer Vision*, Lecture Notes in Computer Science. Springer-Verlag. 1st Iberian Conference on Pattern Recongnition and Image Analysis -IbPRIA2003- Mallorca. Spain. June.

D. Ortiz, I. Garca-Varea, and F. Casacuberta. 2005. Thot: a toolkit to train phrase-based statistical translation models. In *Tenth Machine Translation Summit*, pages 141–148, Phuket, Thailand, September.

J. Tomás and F. Casacuberta. 2001. Monotone statistical translation using word groups. In *Procs. of the Machine Translation Summit VIII*, pages 357–361, Santiago de Compostela, Spain.

J. Tomás and F. Casacuberta. 2003. Combining phrase-based and template-based models in statistical machine translation. In *Pattern Recognition and Image Analisys*, volume 2652 of *LNCS*, pages 1021–1031. Springer-Verlag. 1st bPRIA.

Ye-Yi Wang and Alex Waibel. 1998. Fast decoding for statistical machine translation. In *Proc. of the Int. Conf. on Speech and Language Processing*, pages 1357–1363, Sydney, Australia, November.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. of the 39th Annual Meeting of ACL*, pages 523–530, Toulouse, France, July.

R. Zens, F.J. Och, and H. Ney. 2002. Phrase-based statistical machine translation. In *Advances in artificial intelligence. 25. Annual German Conference on AI*, volume 2479 of *Lecture Notes in Computer Science*, pages 18–32. Springer Verlag, September.