INCREMENTAL TRANSLATION USING HIERARCHICHAL PHRASE-BASED TRANSLATION SYSTEM

Maryam Siahbani, Ramtin Mehdizadeh Seraj, Baskaran Sankaran, Anoop Sarkar

Simon Fraser University
School of Computing Science
Burnaby BC. CANADA
{msiahban,rmehdiza,baskaran,anoop}@cs.sfu.ca

Abstract

Hierarchical phrase-based machine translation [1] (Hiero) is a prominent approach for Statistical Machine Translation usually comparable to or better than conventional phrase-based systems. But Hiero typically uses the CKY decoding algorithm which requires the entire input sentence before decoding begins, as it produces the translation in a bottom-up fashion. Left-to-right (LR) decoding [2] is a promising decoding algorithm for Hiero that produces the output translation in left to right order. In this paper we focus on simultaneous translation using the Hiero translation framework. In simultaneous translation, translations are generated incrementally as source language speech input is processed. We propose a novel approach for incremental translation by integrating segmentation and decoding in LR-Hiero. We compare two incremental decoding algorithms for LR-Hiero and present translation quality scores (BLEU) and the latency of generating translations for both decoders on audio lectures from the TED collection.

Index Terms— Statistical Machine Translation (SMT), Incremental Decoding, Hierarchical Phrase-based Translation (Hiero), Left-to-Right Decoding

1. INTRODUCTION

Hierarchical phrase-based translation (*Hiero*) [3] is a prominent approach for SMT as it is a simple, yet powerful machine translation model. Hiero models encode the translation correspondences in *hierarchical* phrases, unlike the phrase-based models that use contiguous translation phrases. The notion of hierarchy allows the Hiero models to capture long-distance reordering between source and target languages unlike phrase-based models. Additionally they also model discontiguous translations, e.g. translating the English word *not* as *ne* ___ *pas* in French (with an appropriate verb form inserted between *ne* and *pas*). These properties make Hiero models more appropriate for some language pairs than phrase-based models [4, 5, 6].

Hiero uses a lexicalized synchronous context-free grammar (SCFG) extracted from word and phrase alignments of a bitext. Typically, Hiero uses a CKY-style decoding algorithm with time complexity $O(n^3)$ where the source input has n words. Computing the language model score for each hypothesis within CKY decoding requires two histories, the left and the right edge of each span, due to the fact that the target side is built bottom-up from sub-spans.

Typically decoders in statistical machine translation (SMT) make use of language models to ensure that the output is grammatically correct sentence, hence computing the language model score is a crucial part of the process, but an expensive one as well.

The size of a Hiero SCFG grammar is typically larger than phrase-based models extracted from the same data creating challenges in rule extraction and decoding time especially for larger datasets [7].

[2] proposed left-to-right (LR) decoding for Hiero (LR-Hiero henceforth). It is a beam-search decoder which runs in time $O(n^2b)$ in practice where n is the length of source sentence and b is the size of beam [8]. LR-Hiero generates the target side translation left-to-right, by restricting the grammar to SCFG rules which are prefix-lexicalized or in so-called Greibach Normal Form (GNF) on target side. This constraints drastically reduces the size of grammar for LR-Hiero in comparison to Hiero grammars [9]. Restricting the target generation to be left-to-right requires a single language model (LM) history for each hypothesis and this helps speed up the decoding process considerably.

Recently an augmented version of LR decoding has been proposed which is able to address some limitations in terms of translation quality and time efficiency. It shows that LR decoding is a viable alternative to the usual CKY decoding algorithm for Hiero on different language pairs like Czech-English, Chinese-English and German-English [9, 10].

In simultaneous translation, the output translation should be generated incrementally as the user speaks the source language input instead of waiting for the entire sentence. Previous translation services proposed for real-time translation environments, are mainly phrase-based [11, 12, 13, 14, 15]. Since a phrase-based decoder generates translations in a left-to-right manner, it is more suited than the CKY based decoding algorithm used in Hiero decoders which requires the entire input sentence before generating the translation.

We propose to use LR-Hiero for simultaneous translation. LR-Hiero uses a beam-search decoding algorithm and generates the translation in left-to-right manner. In simultaneous translation, the delay between a source language chunk and its translation should be minimal. Previous research on simultaneous translation reduces the task to splitting the input into appropriate segments and then incrementally translating those segments [13, 14, 15]. Like [14], we train a segmentation model that exploits the alignment structure between the source and target languages. We investigate different sets of features for this segmentation task. The trained segmentation model is integrated with an LR-Hiero translation system. We experiment with two different decoding strategies and evaluate our incremental translation system on the speech translation of TED talks from English to French.

2. LEFT-TO-RIGHT HIERO

LR-Hiero uses a constrained lexicalized SCFG which we call a GNF grammar: $X \rightarrow \langle \gamma, \bar{b}\, \beta \rangle$, where X is a non-terminal, γ is a string of non-terminal and terminal symbols, $\bar{b}$ is a string of terminal symbols and β is a possibly empty sequence of non-terminals. This ensures that as each rule is used in a derivation, the target side is generated from left to right. The rules are obtained from a word and phrase aligned bitext using the rule extraction algorithms in [2, 16].

LR-Hiero decoding uses a top-down depth-first search, which strictly grows the hypotheses in target surface ordering. Search on the source side follows an Earley-style search [17], the dot jumps around on the source side of the rules based on the order of nonterminals on the target side. This search is integrated with beam search or cube pruning to find the k-best translations.

Algorithm 1 shows the pseudocode for LR-Hiero decoding with cube pruning [3] (CP). LR-Hiero with CP was introduced in [9].

Each source side non-terminal is instantiated with the legal spans given the input source string, e.g. if there is a Hiero rule $\langle aX_1, a'X_1 \rangle$ and if a only occurs at position 3 in the input, then source side can be matched to span $[3, i]$ for all i, $4 < i \leq n$, where n is length of input. Fig. 1 shows a worked out example of how the decoder works.

Each partial hypothesis h is a 4-tuple (h_t, h_s, h_{cov}, h_c): a translation prefix h_t, a (LIFO-ordered) *list* h_s of uncovered spans, source words coverage set h_{cov} and the hypothesis cost h_c which includes future cost and a score computed based on

Algorithm 1 LR-Hiero Decoding with CP

```
 1: Input sentence: $\mathbf{f} = f_0 f_1 \dots f_n$
 2: $\mathcal{F}$ = FutureCost($\mathbf{f}$)        (Precompute future cost[1] for spans)
 3: $S_0 = \{\}$                              (Create empty initial stack)
 4: $h_0 = (\langle s \rangle, [[0, n]], \emptyset, \mathcal{F}_{[0,n]})$       (Initial hypothesis 4-tuple)
 5: Add $h_0$ to $S_0$               (Push initial hyp into first Stack)
 6: for $i = 1, \dots, n$ do
 7:     $cubeList = \{\}$                        (MRL is max rule length)
 8:     for $p = max(i - \text{MRL}, 0), \dots, i - 1$ do
 9:         $\{G\}$ = Grouped($S_p$)   (based on the first uncovered span)
10:         for $g \in \{G\}$ do
11:             $[u, v] = g_{span}$
12:             R = GetSpanRules($[u, v]$)
13:             for $R_s \in R$ do
14:                 $cube = [g_{hyps}, R_s]$
15:                 Add $cube$ to $cubeList$
16:     $S_i$ = Merge($cubeList, \mathcal{F}$)      (Create stack $S_i$ and add new
        hypotheses to it)
17: return $\arg\min h \in S_n h_c$

18: Merge($CubeList, \mathcal{F}$)
19:     $heapQ = \{\}$
20:     for each $(H, R)$ in $cubeList$ do
21:         $h'$ = getBestHypotheses($(H, R), \mathcal{F}$)   (best hypotheses of
    cubes)
22:         push($heapQ, (h'_c, h', [H, R])$)          (Push new hyp in the
    queue)
23:     $hypList = \{\}$
24:     while $|heapQ| > 0$ and $|hypList| < K$ do
25:         $(h'_c, h', [H, R])$ = pop($heapQ$)  (pop the best hypothesis)
26:         push($heapQ, GetNeighbours([H, R])$)                 (Push
    neighbours to queue)
27:         Add $h'$ to $hypList$
28:     return $hypList$
```

feature values (using a log-linear model). The initial hypothesis is a null string with just a sentence-initial marker $\langle s \rangle$ and the list h_s containing a span of the whole sentence, $[0, n]$. The hypotheses are stored in stacks $S_0, \dots, S_n$, where S_p contains hypotheses covering p source words, just like in stack decoding for phrase-based SMT [19].

To fill stack S_i we consider hypotheses in each stack S_p (for all p, $p < i$)[2], which are first partitioned into a set of groups $\{G\}$, based on their first uncovered span (line 9). Each group g is a 2-tuple (g_{span}, g_{hyps}), where g_{hyps} is a list of hypotheses which share the same first uncovered span g_{span}. *GetSpanRules* finds rules matching the span g_{span}. Cubes are created from each pair of g_{hyps} and possible R_s, which are added to $cubeList$.

The *Merge* routine gets the best hypotheses from all cubes. $GetBestHypotheses((H, R), \mathcal{F})$ produces new hypotheses using current hypothesis H and rule R. The first best hypothesis, h' along with its score h'_c and corresponding

[1]The future cost is precomputed in a way similar to the phrase-based models [18] using only the terminal rules of the grammar.

[2]As the length of rules are limited (at most MRL), we can ignore stacks with index less than i − MRL.

rules	hypotheses $\langle h_t, h_s, h_{cov}, h_c \rangle$
	$\langle$<s>$, [\![[0,8]]\!], --------, 0 \rangle$
G 1) $X \rightarrow \langle$ *schuler* X_1 / *students* $X_1 \rangle$	$\langle$<s> *students* $, [\![[1,8]]\!], *-------, 3.5 \rangle$
G 2) $X \rightarrow \langle X_1$ *heban* X_2 / *have* $X_1 X_2 \rangle$	$\langle$<s> *students have* $, [\![[1,6][7,8]]\!], *------*-, 5.7 \rangle$
3) $X \rightarrow \langle X_1$ *noch nicht* X_2 / *not yet* $X_2 X_1 \rangle$	$\langle$<s> *students have not yet* $, [\![[5,6][1,3][7,8]]\!], *--**-*-, 10.2 \rangle$
4) $X \rightarrow \langle$ *gemacht* / *done* $\rangle$	$\langle$<s> *students have not yet done* $, [\![[1,3][7,8]]\!], *--****-, 12.4 \rangle$
5) $X \rightarrow \langle$ *ihre arbeit* / *their work* $\rangle$	$\langle$<s> *students have not yet done their work* $, [\![[7,8]]\!], *******-, 15.1 \rangle$
6) $X \rightarrow \langle . / . \rangle$	$\langle$<s> *students have not yet done their work .* </s> $, [\![ ]\!], ********, 15.6 \rangle$

Fig. 1. The process of translating a German-English sentence pair in LR-Hiero. Word alignment is shown in Fig. 2. Left side shows the rules used in the derivation (G indicates glue rules as defined in [2]). The hypotheses column shows 4-tuple partial hypotheses: the translation prefix, h_t, the ordered list of yet-to-be-covered spans, h_s, source word coverage vector, h_{cov} and cost h_c (cost includes future cost and hypothesis cost, but we just show hypothesis cost in this figure).

cube (H, R) is placed in a priority queue $heapQ$ and (line 22 in Algorithm 1). The K best hypotheses in the queue are iteratively popped (line 25) and for each hypothesis its neighbours in the cube are pushed to the priority queue (line 26). Decoding process finishes when stack S_n has been filled.

3. SEGMENTATION AND INCREMENTAL TRANSLATION

In incremental translation, the entire input sentence is not given, but we observe it word by word (or segment by segment). For each input sentence $\mathbf{f} = \langle f_1 ... f_n \rangle$ different possible segmentations exist. We seek an optimal segmentation so that segments can be translated to the target language monotonically. In order to do this effectively, we should split the input sentence into a segments that reordering occurs inside each segment and not across segments.

To tackle the segmentation task, we use a classifier to find the segment boundaries. Given the input sentence word by word we apply the classifier to decide whether that word is a segment boundary or not. Recognizing a segment boundary in the input, the decoder then produces the translation chunk for that segment.

To prepare the training data, we extract monotonic phrase alignments using the word alignment produced by GIZA++. Fig. 2 shows word alignment matrix and monotonic phrase alignment for a German-English sentence pair. We run GIZA++ over the entire set of parallel sentences in our training data, but we only use a portion of these parallel sentences in order to extract training examples for training our segmentation model (see section 4 for details). The training set is restricted to segments of length at least 4.

For classification, we use a log-linear model trained on our training set based on different groups of features. Basic features, used in [14], are: words that are at the candidate segment boundary, the position of the boundary in the sentence, and the length of candidate segment. We use this model as our baseline. We use additional features which can be partitioned to two groups:

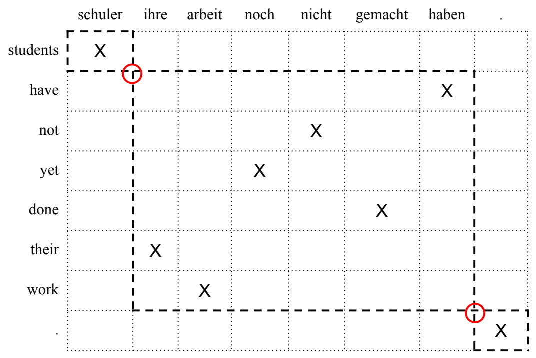

Fig. 2. Word alignment matrix for a German-English sentence pair. Monotone phrases are shown in dashed lines and segment boundaries with red circles.

The first group uses Part Of Speech (POS) tags of the candidate segment as features. We considered the last three POS tags in a segment and also bigrams and trigrams of the POS tags inside each segment.

We hypothesize that the previous state of the decoder might have useful information. This was the motivation for the next group of features. These features can be seen as feedback from decoder about its output. Feedback from decoder includes: language model score (lm), translation probabilities $p(e|f)$ and $p(f|e)$ and two lexically-weighted translation probabilities (tm_0,tm_1,tm_2,tm_3), and the model score (c). We normalize these values and use them as features in segmentation model. Table 1 compares results of using different groups of features on the test set for the segmentation task (10% of the training data is used as test set). In row 4 we use only the lm, tm_0 (which is $p(e|f)$) and c (score) from the decoder.

features	P	R	F1
Basic	0.77	**0.86**	0.81
+ POS	0.7924	**0.84151**	**0.8162**
+ Decoder (all)	0.7971	0.8295	0.8129
+ Decoder (lm,tm_0,c)	**0.8085**	0.8137	0.8110
+ POS + Decoder (lm,tm_0,c)	0.8041	0.8284	**0.8161**
+ POS + Decoder (all)	0.8084	0.8137	0.8110

Table 1. Results on the dev set using different groups of features for segmentation model.

	BLEU	**Time (sec)**
LR-Decoder1	25.72	31.06
LR-Decoder2	24.48	0.84
LR-Hiero (no segmentation)	25.72	19.62

Table 2. Translation quality (BLEU) and latency for translating English-French.

4. EXPERIMENTS

Following the *International Workshop on Spoken Language Translation* (IWSLT) shared task, we would like to evaluate our incremental translation system on the speech translation of TED talks for English-French. We use development (dev2010) and test data (tst2010) of IWSLT 2010. We use the parallel text provided as training data of IWSLT 2011 and Europarl (v7) as the training data for our translation task (about 2M sentence pairs). The training data from IWSLT 2011 is used as the training set for segmentation model (90% as training and 10% as test set).

Table 1 shows the results of the segmentation task on corresponding test set. We use model trained with Basic+POS features as our segmentation model. It outperforms model with basic features in terms of precision and F1 measure and has comparable results (F1 measure) to other models, while it is faster, other models during decoding time (other models need the feedback from decoder to extract features, therefore it requires to run the decoder on the corresponding input chunk, given each input word).

Typical Hiero rule extraction excludes phrase-pairs with unaligned words on boundaries (loose phrases). We use similar rule extraction as Hiero, except that we exclude non-GNF rules and include loose phrase-pairs as terminal rules.

We use a 4-gram language model trained on the WMT2011 corpus (Europarl, News Commentary and UN documents) and use KenLM [20]. We tune weights by minimizing BLEU loss on the devset through MERT [21] and report BLEU scores on the test set (tst2010). We use pop limit 500 for LR-Hiero. In these experiments, we use the reference transcript of the utterance for dev and test sets.

We compare the results in terms of translation quality (BLEU) and latency (of generating partial translations). We concatenate all 11 test sets and report the results on the whole test set. Two incremental decoding strategies using LR-Hiero are used for translation. In both cases we integrate LR-Hiero with the segmentation model: for each input word it queries segmentation model. In our first strategy, LR-Decoder1, if the word is at a segment boundary then LR-Hiero decodes the entire input (from the beginning) and produces the translation so far as the output. In our second decoding strategy, LR-Decoder2, when a segment boundary is recognized, the decoder translates just the last source segment, produces the the translation output for that segment and then updates the last state of the decoder to the current best hypothesis.

LR-Decoder1 changes its output as it receives more input and the final translation output is the same as applying the LR-Hiero decoder over the entire input sentence. While in LR-Decoder2, the translation output is updated over time by adding the translation of later input segments (the decoder does not change the output as it proceeds), which is more appropriate for speech translation.

Table 2 compares the two decoding strategies. The BLEU scores are computed on the final output. LR-Decoder1 obtains a better BLEU score, which is not surprising because the final output is identical to the output of the regular decoder on the entire input sentence. The second column of Table 2 shows the average speed of translating input segments (in terms of seconds). The latency is calculated as the total time taken to translate the whole sentence divided by number of segments. We compute the average over 50 sentences randomly selected from test set. The last row in table 2 shows the results of the regular translation strategy (with no segmentation employed). For a relatively small loss of 1.24 in the BLEU score we obtain a much faster incremental translation system.

5. CONCLUSION

In this paper we proposed the use of LR-Hiero for simultaneous translation for the first time. We trained a segmentation model to split the input to monotone segments to achieve a fast and accurate simultaneous translation. We investigated different sets of features for segmentation task. We obtained a very fast simultaneous translation system (23 times faster than regular translation system) with reasonable translation quality (just 1.24 BLEU score loss).

As future work we are interested to compare our translation framework with other simultaneous MT systems and also apply it on complex word reordering language pairs like Chinese-English. We would like to improve the segmentation model and use it in regular translation tasks to achieve faster translation systems with comparable translation quality.

Acknowledgments This work was partially supported by NSERC, Canada RGPIN: 262313 and RGPAS: 446348. Thanks to the anonymous reviewers for their comments.

6. REFERENCES

[1] David Chiang, "A hierarchical phrase-based model for statistical machine translation," in *In ACL*, 2005, pp. 263–270.

[2] Taro Watanabe, Hajime Tsukada, and Hideki Isozaki, "Left-to-right target generation for hierarchical phrase-based translation," in *Proc. of ACL*, 2006.

[3] David Chiang, "Hierarchical phrase-based translation," *Computational Linguistics*, vol. 33, 2007.

[4] Daniel Marcu and William Wong, "A phrase-based, joint probability model for statistical machine translation," in *Proceedings of Empirical Methods in Natural Language Processing-02*. 2002, pp. 133–139, Association for Computational Linguistics.

[5] Franz Josef Och and Hermann Ney, "Discriminative training and maximum entropy models for statistical machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. 2002, Association for Computational Linguistics.

[6] Franz Josef Och and Hermann Ney, "The alignment template approach to statistical machine translation," *Computational Linguistics*, vol. 30, 2004.

[7] Baskaran Sankaran, Majid Razmara, and Anoop Sarkar, "Kriya - an end-to-end hierarchical phrase-based mt system," *The Prague Bulletin of Mathematical Linguistics (PBML)*, vol. 97, no. 97, pp. 83–98, apr 2012.

[8] Liang Huang and Haitao Mi, "Efficient incremental decoding for tree-to-string translation," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, October 2010, pp. 273–283, Association for Computational Linguistics.

[9] Maryam Siahbani, Baskaran Sankaran, and Anoop Sarkar, "Efficient left-to-right hierarchical phrase-based translation with improved reordering," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, USA, October 2013, Association for Computational Linguistics.

[10] Maryam Siahbani and Anoop Sarkar, "Two improvements to left-to-right decoding for hierarchical phrase-based machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, October 2014, Association for Computational Linguistics.

[11] Christian Fügen, Alex Waibel, and Muntsin Kolss, "Simultaneous translation of lectures and speeches," *Machine Translation*, vol. 21, no. 4, pp. 209–252, 2007.

[12] Baskaran Sankaran, Ajeet Grewal, and Anoop Sarkar, "Incremental decoding for phrase-based statistical machine translation," in *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, Stroudsburg, PA, USA, 2010, WMT '10, pp. 216–223, Association for Computational Linguistics.

[13] Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez, "Real-time incremental speech-to-speech translation of dialogs," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Stroudsburg, PA, USA, 2012, NAACL HLT '12, pp. 437–445, Association for Computational Linguistics.

[14] Mahsa Yarmohammadi, Vivek K Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran, "Incremental segmentation and decoding strategies for simultaneous translation," in *Proceedings of The 6th International Joint Conference on Natural Language Processing*. 2013, Association for Computational Linguistics.

[15] Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura, "Optimizing segmentation strategies for simultaneous speech translation," in *The 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Baltimore, USA, June 2014.

[16] Maryam Siahbani and Anoop Sarkar, "Expressive hierarchical rule extraction for left-to-right translation," in *Proceedings of the 11th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-2014).*, Vancouver, Canada, October 2014, Association for Computational Linguistics.

[17] Jay Earley, "An efficient context-free parsing algorithm," *Commun. ACM*, vol. 13, no. 2, pp. 94–102, Feb. 1970.

[18] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst, "Moses: open source toolkit for statistical machine translation," in *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, Stroudsburg, PA, USA, 2007, ACL '07, pp. 177–180, Association for Computational Linguistics.

[19] Philipp Koehn, Franz Josef Och, and Daniel Marcu, "Statistical phrase-based translation," in *Proc. of NAACL*, 2003.

[20] Kenneth Heafield, "KenLM: Faster and smaller language model queries," in *In Proc. of the Sixth Workshop on Statistical Machine Translation*, 2011.

[21] Franz Josef Och, "Minimum error rate training in statistical machine translation," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, Stroudsburg, PA, USA, 2003, ACL '03, pp. 160–167, Association for Computational Linguistics.