

Hierarchical Phrase-Based Translation with Weighted Finite State Transducers

Gonzalo Iglesias* Adrià de Gispert[‡] Eduardo R. Banga* William Byrne[‡]

* University of Vigo. Dept. of Signal Processing and Communications. Vigo, Spain

{giglesia, erbang}@gts.tsc.uvigo.es

[‡] University of Cambridge. Dept. of Engineering. CB2 1PZ Cambridge, U.K.

{ad465, wjb31}@eng.cam.ac.uk

To appear in proceedings of NAACL-HLT 2009

Abstract

This paper describes a lattice-based decoder for hierarchical phrase-based translation. The decoder is implemented with standard WFST operations as an alternative to the well-known cube pruning procedure. We find that the use of WFSTs rather than k-best lists requires less pruning in translation search, resulting in fewer search errors, direct generation of translation lattices in the target language, better parameter optimization, and improved translation performance when rescoring with long-span language models and MBR decoding. We report translation experiments for the Arabic-to-English and Chinese-to-English NIST translation tasks and contrast the WFST-based hierarchical decoder with hierarchical translation under cube pruning.

1 Introduction

Hierarchical phrase-based translation generates translation hypotheses via the application of hierarchical rules in CYK parsing (Chiang, 2005). *Cube pruning* is used to apply language models at each cell of the CYK grid as part of the search for a k-best list of translation candidates (Chiang, 2005; Chiang, 2007). While this approach is very effective and has been shown to produce very good quality translation, the reliance on k-best lists is a limitation. We take an alternative approach and describe a lattice-based hierarchical decoder implemented with Weighted Finite State Transducers (WFSTs). In every CYK cell we build a single, minimal word lattice containing all possible translations of the source sentence span covered by that cell. When derivations

contain non-terminals, we use pointers to lower-level lattices for memory efficiency. The pointers are only expanded to the actual translations if pruning is required during search; expansion is otherwise only carried out at the upper-most cell, after the full CYK grid has been traversed.

We describe how this decoder can be easily implemented with WFSTs. For this we employ the OpenFST libraries (Allauzen et al., 2007). Using standard FST operations such as composition, epsilon removal, determinization, minimization and shortest-path, we find this search procedure to be simpler to implement than cube pruning. The main modeling advantages are a significant reduction in search errors, a simpler implementation, direct generation of target language word lattices, and better integration with other statistical MT procedures. We report translation results in Arabic-to-English and Chinese-to-English translation and contrast the performance of lattice-based and cube pruning hierarchical decoding.

1.1 Related Work

Hierarchical phrase-based translation has emerged as one of the dominant current approaches to statistical machine translation. Hiero translation systems incorporate many of the strengths of phrase-based translation systems, such as feature-based translation and strong target language models, while also allowing flexible translation and movement based on hierarchical rules extracted from aligned parallel text. We summarize some extensions to the basic approach to put our work in context.

Hiero Search Refinements Huang and Chiang (2007) offer several refinements to cube pruning to improve translation speed. Venugopal et al. (2007) introduce a Hiero variant with relaxed constraints for hypothesis recombination during parsing; speed and results are comparable to those of cube pruning, as described by Chiang (2007). Li and Khudanpur (2008) report significant improvements in translation speed by taking unseen n-grams into account within cube pruning to minimize language model requests. Dyer et al. (2008) extend the translation of source sentences to translation of input lattices following Chappelier et al. (1999).

Extensions to Hiero Several authors describe extensions to Hiero, to incorporate additional syntactic information (Zollmann and Venugopal, 2006; Zhang and Gildea, 2006; Shen et al., 2008; Marton and Resnik, 2008), or to combine it with discriminative latent models (Blunsom et al., 2008).

Analysis and Contrastive Experiments Zollman et al. (2008) compare phrase-based, hierarchical and syntax-augmented decoders for translation of Arabic, Chinese, and Urdu into English. Lopez (2008) explores whether lexical reordering or the phrase discontinuity inherent in hierarchical rules explains improvements over phrase-based systems. Hierarchical translation has also been used to great effect in combination with other translation architectures, e.g. (Sim et al., 2007; Rosti et al., 2007).

WFSTs for Translation There is extensive work in using Weighted Finite State Transducer for machine translation (Bangalore and Ricciardi, 2001; Casacuberta, 2001; Kumar and Byrne, 2005; Mathias and Byrne, 2006; Graehl et al., 2008).

To our knowledge, this paper presents the first description of hierarchical phrase-based translation in terms of lattices rather than k-best lists. The next section describes hierarchical phrase-based translation with WFSTs, including the lattice construction over the CYK grid and pruning strategies. Section 3 reports translation experiments for Arabic-to-English and Chinese-to-English, and Section 4 concludes.

2 Hierarchical Translation with WFSTs

The translation system is based on a variant of the CYK algorithm closely related to CYK+ (Chappe-

lier and Rajman, 1998). Parsing follows the description of Chiang (2005; 2007), maintaining backpointers and employing hypothesis recombination without pruning. The underlying model is a synchronous context-free grammar consisting of a set $\mathbf{R} = \{R^r\}$ of rules $R^r : N \rightarrow \langle \gamma^r, \alpha^r \rangle / p^r$, with ‘glue’ rules, $S \rightarrow \langle X, X \rangle$ and $S \rightarrow \langle S X, S X \rangle$. If a rule has probability p^r , it is transformed to a cost c^r ; here we use the tropical semiring, so $c^r = -\log p^r$. N denotes a non-terminal; in this paper, N can be either S , X , or V (see section 3.2). \mathbf{T} denotes the terminals (words), and the grammar builds parses based on strings $\gamma, \alpha \in \{\{S, X, V\} \cup \mathbf{T}\}^+$. Each cell in the CYK grid is specified by a non-terminal symbol and position in the CYK grid: (N, x, y) , which spans s_x^{x+y-1} on the source sentence.

In effect, the source language sentence is parsed using a context-free grammar with rules $N \rightarrow \gamma$. The generation of translations is a second step that follows parsing. For this second step, we describe a method to construct word lattices with all possible translations that can be produced by the hierarchical rules. Construction proceeds by traversing the CYK grid along the backpointers established in parsing. In each cell (N, x, y) in the CYK grid, we build a target language word lattice $\mathcal{L}(N, x, y)$. This lattice contains every translation of s_x^{x+y-1} from every derivation headed by N . These lattices also contain the translation scores on their arc weights.

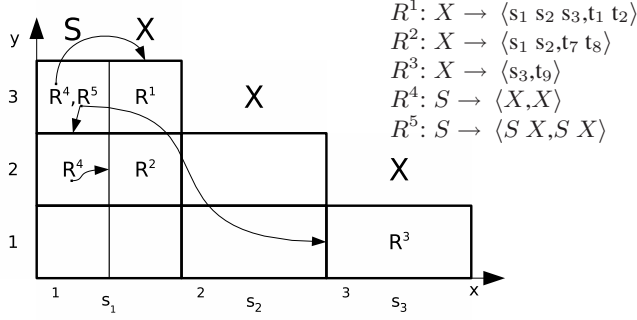
The ultimate objective is the word lattice $\mathcal{L}(S, 1, J)$ which corresponds to all the analyses that cover the source sentence s_1^J . Once this is built, we can apply a target language model to $\mathcal{L}(S, 1, J)$ to obtain the final target language translation lattice (Allauzen et al., 2003).

We use the approach of Mohri (2002) in applying WFSTs to statistical NLP. This fits well with the use of the OpenFST toolkit (Allauzen et al., 2007) to implement our decoder.

2.1 Lattice Construction Over the CYK Grid

In each cell (N, x, y) , the set of rule indices used by the parser is denoted $R(N, x, y)$, i.e. for $r \in R(N, x, y)$, $N \rightarrow \langle \gamma^r, \alpha^r \rangle$ was used in at least one derivation involving that cell.

For each rule $R^r, r \in R(N, x, y)$, we build a lattice $\mathcal{L}(N, x, y, r)$. This lattice is derived from the target side of the rule α^r by concatenating lattices



$$\begin{aligned}
\mathcal{L}(S, 1, 3) &= \mathcal{L}(S, 1, 3, 4) \oplus \mathcal{L}(S, 1, 3, 5) \\
\mathcal{L}(S, 1, 3, 4) &= \mathcal{L}(X, 1, 3) = \mathcal{L}(X, 1, 3, 1) = \\
&= \mathcal{A}(t_1) \otimes \mathcal{A}(t_2) \\
\mathcal{L}(S, 1, 3, 5) &= \mathcal{L}(S, 1, 2) \otimes \mathcal{L}(X, 3, 1) \\
\mathcal{L}(S, 1, 2) &= \mathcal{L}(S, 1, 2, 4) = \mathcal{L}(X, 1, 2) = \\
&= \mathcal{L}(X, 1, 2, 2) = \mathcal{A}(t_7) \otimes \mathcal{A}(t_8) \\
\mathcal{L}(X, 3, 1) &= \mathcal{L}(X, 3, 1, 3) = \mathcal{A}(t_9) \\
\mathcal{L}(S, 1, 3, 5) &= \mathcal{A}(t_7) \otimes \mathcal{A}(t_8) \otimes \mathcal{A}(t_9)
\end{aligned}$$

$$\mathcal{L}(S, 1, 3) = (\mathcal{A}(t_1) \otimes \mathcal{A}(t_2)) \oplus (\mathcal{A}(t_7) \otimes \mathcal{A}(t_8) \otimes \mathcal{A}(t_9))$$

Figure 1: Production of target lattice $\mathcal{L}(S, 1, 3)$ using translation rules within CYK grid for sentence $s_1s_2s_3$. The grid is represented here in two dimensions (x, y) . In practice only the first column accepts both non-terminals (S, X) . For this reason it is divided in two subcolumns.

corresponding to the elements of $\alpha^r = \alpha_1^r \dots \alpha_{|\alpha^r|}^r$. If an α_i^r is a terminal, creating its lattice is straightforward. If α_i^r is a non-terminal, it refers to a cell (N', x', y') lower in the grid identified by the backpointer $BP(N, x, y, r, i)$; in this case, the lattice used is $\mathcal{L}(N', x', y')$. Taken together,

$$\mathcal{L}(N, x, y, r) = \bigotimes_{i=1..|\alpha^r|} \mathcal{L}(N, x, y, r, i) \quad (1)$$

$$\mathcal{L}(N, x, y, r, i) = \begin{cases} \mathcal{A}(\alpha_i) & \text{if } \alpha_i \in \mathbf{T} \\ \mathcal{L}(N', x', y') & \text{else} \end{cases} \quad (2)$$

where $\mathcal{A}(t)$, $t \in \mathbf{T}$ returns a single-arc acceptor which accepts only the symbol t . The lattice $\mathcal{L}(N, x, y)$ is then built as the union of lattices corresponding to the rules in $R(N, x, y)$:

$$\mathcal{L}(N, x, y) = \bigoplus_{r \in R(N, x, y)} \mathcal{L}(N, x, y, r) \quad (3)$$

Lattice union and concatenation are performed using the \oplus and \otimes WFST operations respectively, as described by Allauzen et al.(2007). If a rule R^r has a cost c^r , it is applied to the exit state of the lattice $\mathcal{L}(N, x, y, r)$ prior to the operation of Equation 3.

2.1.1 An Example of Phrase-based Translation

Figure 1 illustrates this process for a three word source sentence $s_1s_2s_3$ under monotone phrase-based translation. The left-hand side shows the state of the CYK grid after parsing using the rules R^1 to R^5 . These include 3 rules with only terminals (R^1 ,

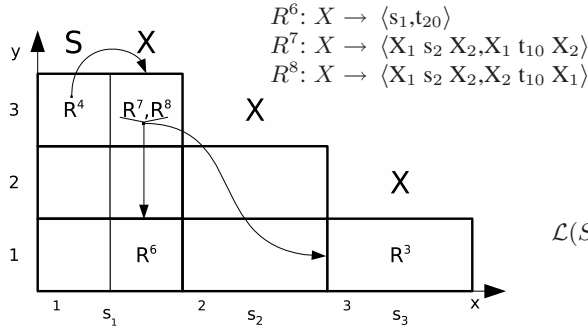
R^2 , R^3) and the glue rules (R^4 , R^5). Arrows represent backpointers to lower-level cells. We are interested in the upper-most S cell $(S, 1, 3)$, as it represents the search space of translation hypotheses covering the whole source sentence. Two rules (R^4 , R^5) are in this cell, so the lattice $\mathcal{L}(S, 1, 3)$ will be obtained by the union of the two lattices found by the backpointers of these two rules. This process is explicitly derived in the right-hand side of Figure 1.

2.1.2 An Example of Hierarchical Translation

Figure 2 shows a hierarchical scenario for the same sentence. Three rules, R^6 , R^7 , R^8 , are added to the example of Figure 1, thus providing two additional derivations. This makes use of sublattices already produced in the creation of $\mathcal{L}(S, 1, 3, 5)$ and $\mathcal{L}(X, 1, 3, 1)$ in Figure 1; these are within $\{\}$.

2.2 A Procedure for Lattice Construction

Figure 3 presents an algorithm to build the lattice for every cell. The algorithm uses memoization: if a lattice for a requested cell already exists, it is returned (line 2); otherwise it is constructed via equations 1,2,3. For every rule, each element of the target side (lines 3,4) is checked as terminal or non-terminal (equation 2). If it is a terminal element (line 5), a simple acceptor is built. If it is a non-terminal (line 6), the lattice associated to its backpointer is returned (lines 7 and 8). The complete lattice $\mathcal{L}(N, x, y, r)$ for each rule is built by equation 1 (line 9). The lattice $\mathcal{L}(N, x, y)$ for this cell is then found by union of all the component rules (line 10, equation 3); this lattice is then reduced by



$$\begin{aligned}
\mathcal{L}(S, 1, 3) &= \mathcal{L}(S, 1, 3, 4) \oplus \{\mathcal{L}(S, 1, 3, 5)\} \\
\mathcal{L}(S, 1, 3, 4) &= \mathcal{L}(X, 1, 3) = \\
&= \{\mathcal{L}(X, 1, 3, 1)\} \oplus \mathcal{L}(X, 1, 3, 7) \oplus \mathcal{L}(X, 1, 3, 8) \\
\mathcal{L}(X, 1, 3, 7) &= \mathcal{L}(X, 1, 1, 6) \otimes \mathcal{A}(t_{10}) \otimes \mathcal{L}(X, 3, 1, 3) = \\
&= \mathcal{A}(t_{20}) \otimes \mathcal{A}(t_{10}) \otimes \mathcal{A}(t_9) \\
\mathcal{L}(X, 1, 3, 8) &= \mathcal{A}(t_9) \otimes \mathcal{A}(t_{10}) \otimes \mathcal{A}(t_{20})
\end{aligned}$$

$$\begin{aligned}
\mathcal{L}(S, 1, 3) &= \{\{\mathcal{A}(t_1) \otimes \mathcal{A}(t_2)\}\} \oplus \\
&\oplus \{\mathcal{A}(t_{20}) \otimes \mathcal{A}(t_{10}) \otimes \mathcal{A}(t_9)\} \oplus \{\mathcal{A}(t_9) \otimes \mathcal{A}(t_{10}) \otimes \mathcal{A}(t_{20})\} \oplus \\
&\oplus \{\{\mathcal{A}(t_7) \otimes \mathcal{A}(t_8) \otimes \mathcal{A}(t_9)\}\}
\end{aligned}$$

Figure 2: Translation as in Figure 1 but with additional rules R^6, R^7, R^8 . Lattices previously derived appear within $\{\}$.

standard WFST operations (lines 11,12,13). It is important at this point to remove any epsilon arcs which may have been introduced by the various WFST union, concatenation, and replacement operations (Allauzen et al., 2007).

```

1 | function buildFst(N,x,y)
2 |   if  $\exists \mathcal{L}(N, x, y)$  return  $\mathcal{L}(N, x, y)$ 
3 |   for  $r \in R(N, x, y)$ ,  $R^r : N \rightarrow \langle \gamma, \alpha \rangle$ 
4 |     for  $i = 1 \dots |\alpha|$ 
5 |       if  $\alpha_i \in \mathbf{T}$ ,  $\mathcal{L}(N, x, y, r, i) = \mathcal{A}(\alpha_i)$ 
6 |       else
7 |          $(N', x', y') = BP(\alpha_i)$ 
8 |          $\mathcal{L}(N, x, y, r, i) = \text{buildFst}(N', x', y')$ 
9 |          $\mathcal{L}(N, x, y, r) = \bigotimes_{i=1 \dots |\alpha|} \mathcal{L}(N, x, y, r, i)$ 
10 |  $\mathcal{L}(N, x, y) = \bigoplus_{r \in R(N, x, y)} \mathcal{L}(N, x, y, r)$ 
11 | fstRmEpsilon  $\mathcal{L}(N, x, y)$ 
12 | fstDeterminize  $\mathcal{L}(N, x, y)$ 
13 | fstMinimize  $\mathcal{L}(N, x, y)$ 
14 | return  $\mathcal{L}(N, x, y)$ 

```

Figure 3: Recursive Lattice Construction.

2.3 Delayed Translation

Equation 2 leads to the recursive construction of lattices in upper-levels of the grid through the union and concatenation of lattices from lower levels. If equations 1 and 3 are actually carried out over fully expanded word lattices, the memory required by the upper lattices will increase exponentially.

To avoid this, we use special arcs that serve as pointers to the low-level lattices. This effectively builds a skeleton of the desired lattice and delays the creation of the final word lattice until a single replacement operation is carried out in the top cell $(S, 1, J)$. To make this exact, we define a function

$g(N, x, y)$ which returns a unique tag for each lattice in each cell, and use it to redefine equation 2. With the backpointer $(N', x', y') = BP(N, x, y, r, i)$, these special arcs are introduced as:

$$\mathcal{L}(N, x, y, r, i) = \begin{cases} \mathcal{A}(\alpha_i) & \text{if } \alpha_i \in \mathbf{T} \\ \mathcal{A}(g(N', x', y')) & \text{else} \end{cases} \quad (4)$$

The resulting lattices $\mathcal{L}(N, x, y)$ are a mix of target language words and lattice pointers (Figure 4, top). However each still represents the entire search space of all translation hypotheses covering the span. Importantly, operations on these lattices – such as lossless size reduction via determinization and minimization – can still be performed. Owing to the existence of multiple hierarchical rules which share the same low-level dependencies, these operations can greatly reduce the size of the skeleton lattice; Figure 4 shows the effect on the translation example. This process is carried out for the lattice at every cell, even at the lowest level where there are only sequences of word terminals. As stated, size reductions can be significant. However not all redundancy is removed, since duplicate paths may arise through the concatenation and union of sublattices with different spans.

At the upper-most cell, the lattice $\mathcal{L}(S, 1, J)$ contains pointers to lower-level lattices. A single FST replace operation (Allauzen et al., 2007) recursively substitutes all pointers by their lower-level lattices until no pointers are left, thus producing the complete target word lattice for the whole source sentence. The use of the lattice pointer arc was inspired by the ‘lazy evaluation’ techniques developed by Mohri et al (2000). Its implementation uses the infrastructure provided by the OpenFST libraries for

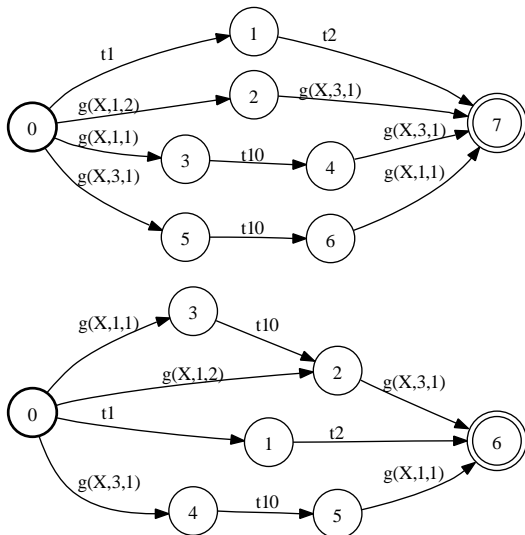


Figure 4: Delayed translation WFST with derivations from Figure 1 and Figure 2 before [t] and after minimization [b].

delayed composition, etc.

2.4 Pruning in Lattice Construction

The final translation lattice $\mathcal{L}(S, 1, J)$ can grow very large after the pointer arcs are expanded. We therefore apply a word-based language model, via WFST composition, and perform likelihood-based pruning (Allauzen et al., 2007) based on the combined translation and language model scores.

Pruning can also be performed on sublattices during search. One simple strategy is to monitor the number of states in the determinized lattices $\mathcal{L}(N, x, y)$. If this number is above a threshold, we expand any pointer arcs and apply a word-based language model via composition. The resulting lattice is then reduced by likelihood-based pruning, after which the LM scores are removed. This search pruning can be very selective. For example, the pruning threshold can depend on the height of the cell in the grid. In this way the risk of search errors can be controlled.

3 Translation Experiments

We report experiments on the NIST MT08 Arabic-to-English and Chinese-to-English translation tasks. We contrast two hierarchical phrase-based decoders. The first decoder, Cube Pruning Hiero (CPH), is a k-

best decoder using cube pruning implemented as described by Chiang (2007). In our implementation, k-best lists contain unique hypotheses. The second decoder, FSTHiero (FSTH), is a lattice-based decoder implemented with Weighted Finite State Transducers as described in the previous section. Hypotheses are generated after determinization under the tropical semiring so that scores assigned to hypotheses arise from single minimum cost / maximum likelihood derivations. We also use a variant of the k-best decoder which works in alignment mode: given an input k-best list, it outputs the feature scores of each hypothesis in the list without applying any pruning. This is used for Minimum Error Training (MET) with the FSTH system.

These two language pairs pose very different translation challenges. For example, Chinese-to-English translation requires much greater word movement than Arabic-to-English. In the framework of hierarchical translation systems, we have found that shallow decoding (see section 3.2) is as good as full hierarchical decoding in Arabic-to-English (Iglesias et al., 2009). In Chinese-to-English, we have not found this to be the case. Therefore, we contrast the performance of FSTH and CPH under shallow hierarchical decoding for Arabic-to-English, while for Chinese-to-English we perform full hierarchical decoding.

Both hierarchical translation systems share a common architecture. For both language pairs, alignments are generated over the parallel data. The following features are extracted and used in translation: target language model, source-to-target and target-to-source phrase translation models, word and rule penalties, number of usages of the glue rule, source-to-target and target-to-source lexical models, and three rule count features inspired by Bender et al. (2007). The initial English language model is a 4-gram estimated over the parallel text and a 965 million word subset of monolingual data from the English Gigaword Third Edition. Details of the parallel corpus and development sets used for each language pair are given in their respective section.

Standard MET (Och, 2003) iterative parameter estimation under IBM BLEU (Papineni et al., 2001) is performed on the corresponding development set. For the CPH system, MET is done following Chiang (2007). For the FSTH system, we obtain a k-best list

from the translation lattice and extract each feature score with the aligner variant of the k-best decoder. After translation with optimized feature weights, we carry out the two following rescoring steps.

- *Large-LM rescoring.* We build sentence-specific zero-cutoff stupid-backoff (Brants et al., 2007) 5-gram language models, estimated using ~ 4.7 B words of English newswire text, and apply them to rescore either 10000-best lists generated by CPH or word lattices generated by FSTH. Lattices provide a vast search space relative to k-best lists, with translation lattice sizes of 10^{81} hypotheses reported in the literature (Tromble et al., 2008).
- *Minimum Bayes Risk (MBR).* We rescore the first 1000-best hypotheses with MBR, taking the negative sentence level BLEU score as the loss function (Kumar and Byrne, 2004).

3.1 Building the Rule Sets

We extract hierarchical phrases from word alignments, applying the same restrictions as introduced by Chiang (2005). Additionally, following Iglesias et al. (2009) we carry out two rule filtering strategies:

- we exclude rules with two non-terminals with the same order on the source and target side
- we consider only the 20 most frequent translations for each rule

For each development set, this produces approximately 4.3M rules in Arabic-to-English and 2.0M rules in Chinese-to-English.

3.2 Arabic-to-English Translation

We translate Arabic-to-English with shallow hierarchical decoding, *i.e.* only phrases are allowed to be substituted into non-terminals. The rules used in this case are, in addition to the glue rules:

$$\begin{aligned} X &\rightarrow \langle \gamma_s, \alpha_s \rangle \\ X &\rightarrow \langle V, V \rangle \\ V &\rightarrow \langle s, t \rangle \\ s, t &\in \mathbf{T}^+; \gamma_s, \alpha_s \in (\{V\} \cup \mathbf{T})^+ \end{aligned}$$

For translation model training, we use all allowed parallel corpora in the NIST MT08 Arabic track

(~ 150 M words per language). In addition to the MT08 set itself, we use a development set *mt02-05-tune* formed from the odd numbered sentences of the NIST MT02 through MT05 evaluation sets; the even numbered sentences form the validation set *mt02-05-test*. The *mt02-05-tune* set has 2,075 sentences.

The cube pruning decoder, CPH, employs k-best lists of depth $k=10000$ (unique). Using deeper lists results in excessive memory and time requirements. In contrast, the WFST-based decoder, FSTH, requires no local pruning during lattice construction for this task and the language model is not applied until the lattice is fully built at the upper-most cell of the CYK grid.

Table 1 shows results for *mt02-05-tune*, *mt02-05-test* and *mt08*, as measured by lowercased IBM BLEU and TER (Snover et al., 2006). MET parameters are optimized for the CPH decoder. As shown in rows ‘a’ and ‘b’, results after MET are comparable.

Search Errors Since both decoders use exactly the same features, we can measure their search errors on a sentence-by-sentence basis. A search error is assigned to one of the decoders if the other has found a hypothesis with lower cost. For *mt02-05-tune*, we find that in 18.5% of the sentences FSTH finds a hypothesis with lower cost than CPH. In contrast, CPH never finds any hypothesis with lower cost for any sentence. This is as expected: the FSTH decoder requires no pruning prior to applying the language model, so search is exact.

Lattice/k-best Quality Rescoring results are different for cube pruning and WFST-based decoders. Whereas CPH improves by 0.9 BLEU, FSTH improves over 1.5 BLEU. Clearly, search errors in CPH not only affect the 1-best output but also the quality of the resulting k-best lists. For CPH, this limits the possible gain from subsequent rescoring steps such as large LMs and MBR.

Translation Speed CPH requires an average of 1.1 seconds per input word. FSTH cuts this time by half, producing output at a rate of 0.5 seconds per word. It proves much more efficient to process compact lattices containing many hypotheses rather than to independently processing each one of them in k-best form.

The mixed case NIST BLEU-4 for the FSTH system on *mt08* is 42.9. This is directly comparable to

decoder		<i>mt02-05-tune</i>		<i>mt02-05-test</i>		<i>mt08</i>	
		BLEU	TER	BLEU	TER	BLEU	TER
a	CPH	52.2	41.6	51.5	42.2	42.5	48.6
	+5gram	53.1	41.0	52.5	41.5	43.3	48.3
	+MBR	53.2	40.8	52.6	41.4	43.4	48.1
b	FSTH	52.2	41.5	51.6	42.1	42.4	48.7
	+5gram	53.3	40.6	52.7	41.3	43.7	48.1
	+MBR	53.7	40.4	53.3	40.9	44.0	48.0

Decoding time in secs/word: 1.1 for CPH; 0.5 for FSTH.

Table 1: Constrative Arabic-to-English translation results (lower-cased IBM BLEU | TER) after MET and subsequent rescoring steps. Decoding time reported for *mt02-05-tune*.

the official MT08 Constrained Training Track evaluation results¹.

3.3 Chinese-to-English Translation

We translate Chinese-to-English with full hierarchical decoding, *i.e.* hierarchical rules are allowed to be substituted into non-terminals. We consider a maximum span of 10 words for the application of hierarchical rules and only glue rules are allowed at upper levels of the CYK grid.

For translation model training, we use all available data for the GALE 2008 evaluation², approx. 250M words per language. In addition to the MT08 set itself, we use a development set *tune-nw* and a validation set *test-nw*. These contain a mix of the newswire portions of MT02 through MT05 and additional developments sets created by translation within the GALE program. The *tune-nw* set has 1,755 sentences.

Again, the CPH decoder employs k-best lists of depth $k=10000$. The FSTH decoder applies pruning in search as described in Section 2.4, so that any lattice in the CYK grid is pruned if it covers at least 3 source words and contains more than 10k states. The likelihood pruning threshold relative to the best path in the lattice is 9. This is a very broad threshold so that very few paths are discarded.

Improved Optimization Table 2 shows results for *tune-nw*, *test-nw* and *mt08*, as measured by lower-

cased IBM BLEU and TER. The first two rows show results for CPH when using MET parameters optimized over k-best lists produced by CPH (row ‘a’) and by FSTH (row ‘b’). We find that using the k-best list obtained by the FSTH decoder yields better parameters during optimization. Tuning on the FSTH k-best lists improves the CPH BLEU score, as well. We find consistent improvements in BLEU; TER also improves overall, although less consistently.

Search Errors Measured over the *tune-nw* development set, FSTH finds a hypothesis with lower cost in 48.4% of the sentences. In contrast, CPH never finds any hypothesis with a lower cost for any sentence, indicating that the described pruning strategy for FSTH is much broader than that of CPH. Note that CPH search errors are more frequent for this language pair. This is due to the larger search space required in fully hierarchical translation; the larger the search space, the more search errors will be produced by the cube pruning k-best implementation.

Lattice/k-best Quality The lattices produced by FSTH yield greater gains in LM rescoring than the k-best lists produced by CPH. Including the subsequent MBR rescoring, translation improves as much as 1.2 BLEU, compared to 0.7 BLEU with CPH. The mixed case NIST BLEU-4 for the FSTH system on *mt08* is 27.8, comparable to official results in the UnConstrained Training Track of the NIST 2008 evaluation.

4 Conclusions

The lattice-based decoder for hierarchical phrase-based translation described in this paper can be easily implemented using Weighted Finite State Trans-

¹Full MT08 results are available at http://www.nist.gov/speech/tests/mt/2008/doc/mt08_official_results_v0.html. It is worth noting that many of the top entries make use of system combination; the results reported here are for single system translation.

²See <http://projects ldc.upenn.edu/gale/data/catalog.html>.

decoder		MET k-best	<i>tune-nw</i>		<i>test-nw</i>		<i>mt08</i>	
			BLEU	TER	BLEU	TER	BLEU	TER
a	CPH	CPH	31.6	59.7	31.9	59.7	–	–
b	CPH	FSTH	31.7	60.0	32.2	59.9	27.2	60.2
	+5gram		32.2	59.3	32.6	59.4	27.8	59.3
	+MBR		32.4	59.2	32.7	59.4	28.1	59.3
c	FSTH	FSTH	32.0	60.1	32.2	60.0	27.1	60.5
	+5gram		32.7	58.3	33.1	58.4	28.1	59.1
	+MBR		32.9	58.4	33.4	58.5	28.9	58.9

Table 2: Contrastive Chinese-to-English translation results (lower-cased IBM BLEU|TER) after MET and subsequent rescoring steps. The MET k-best column indicates which decoder generated the k-best lists used in MET optimization.

ducers. We find many benefits in this approach to translation. From a practical perspective, the computational operations required can be easily carried out using standard operations already implemented in general purpose libraries. From a modeling perspective, the compact representation of multiple translation hypotheses in lattice form requires less pruning in hierarchical search. The result is fewer search errors and reduced overall memory use relative to cube pruning over k-best lists. We also find improved performance of subsequent rescoring procedures which rely on the translation scores. In direct comparison to k-best lists generated under cube pruning, we find that MET parameter optimization, rescoring with large language models, and MBR decoding, are all improved when applied to translations generated by the lattice-based hierarchical decoder.

Acknowledgments

This work was supported in part by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022. G. Iglesias supported by Spanish Government research grant BES-2007-15956 (project TEC2006-13694-C03-03).

References

- Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of ACL*, pages 557–564.
- Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A
- general and efficient weighted finite-state transducer library. In *Proceedings of CIAA*, pages 11–23.
- Srinivas Bangalore and Giuseppe Riccardi. 2001. A finite-state approach to machine translation. In *Proceedings of NAACL*.
- Oliver Bender, Evgeny Matusov, Stefan Hahn, Sasa Hasan, Shahram Khadivi, and Hermann Ney. 2007. The RWTH Arabic-to-English spoken language translation system. In *Proceedings of ASRU*, pages 396–401.
- Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-HLT*, pages 200–208.
- Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP-ACL*, pages 858–867.
- Francisco Casacuberta. 2001. Finite-state transducers for speech-input translation. In *Proceedings of ASRU*.
- Jean-Cédric Chappelier and Martin Rajman. 1998. A generalized CYK algorithm for parsing stochastic CFG. In *Proceedings of TAPD*, pages 133–137.
- Jean-Cédric Chappelier, Martin Rajman, Ramón Aragiés, and Antoine Rozenknop. 1999. Lattice parsing for speech recognition. In *Proceedings of TALN*, pages 95–104.
- David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270.
- David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.
- Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL-HLT*, pages 1012–1020.
- Jonathan Graehl, Kevin Knight, and Jonathan May. 2008. Training tree transducers. *Computational Linguistics*, 34(3):391–427.

- Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*, pages 144–151.
- Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009. Rule filtering by pattern for efficient hierarchical translation. In *Proceedings of EACL*.
- Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of HLT-NAACL*, pages 169–176.
- Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of HLT-EMNLP*, pages 161–168.
- Zhifei Li and Sanjeev Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of the ACL-HLT Second Workshop on Syntax and Structure in Statistical Translation*, pages 10–18.
- Adam Lopez. 2008. Tera-scale translation models via pattern matching. In *Proceedings of COLING*, pages 505–512.
- Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-HLT*, pages 1003–1011.
- Lambert Mathias and William Byrne. 2006. Statistical phrase-based speech translation. In *Proceedings of ICASSP*.
- Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231:17–32.
- Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. In *Computer Speech and Language*, volume 16, pages 69–88.
- Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.
- Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007. Combining outputs from multiple machine translation systems. In *Proceedings of HLT-NAACL*, pages 228–235.
- Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-HLT*, pages 577–585.
- Khe Chai Sim, William Byrne, Mark Gales, Hichem Sahbi, and Phil Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *Proceedings of ICASSP*, volume 4, pages 105–108.
- Matthew Snover, Bonnie J. Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*, pages 223–231.
- Roy Tromble, Shankar Kumar, Franz J. Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *Proceedings of EMNLP*, pages 620–629.
- Ashish Venugopal, Andreas Zollmann, and Vogel Stephan. 2007. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *Proceedings of HLT-NAACL*, pages 500–507.
- Hao Zhang and Daniel Gildea. 2006. Synchronous binarization for machine translation. In *Proceedings of HLT-NAACL*, pages 256–263.
- Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of NAACL Workshop on Statistical Machine Translation*, pages 138–141.
- Andreas Zollmann, Ashish Venugopal, Franz Och, and Jay Ponte. 2008. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of COLING*, pages 1145–1152.