Hierarchical Phrase-Based Translation with Weighted Finite-State Transducers and Shallow-*n* Grammars

Adrià de Gispert[*]
University of Cambridge

Gonzalo Iglesias[**]
University of Vigo

Graeme Blackwood[*]
University of Cambridge

Eduardo R. Banga[**]
University of Vigo

William Byrne[*]
University of Cambridge

In this article we describe HiFST, a lattice-based decoder for hierarchical phrase-based translation and alignment. The decoder is implemented with standard Weighted Finite-State Transducer (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, better parameter optimization, and improved translation performance. The direct generation of translation lattices in the target language can improve subsequent rescoring procedures, yielding further gains when applying long-span language models and Minimum Bayes Risk decoding. We also provide insights as to how to control the size of the search space defined by hierarchical rules. We show that shallow-n grammars, low-level rule catenation, and other search constraints can help to match the power of the translation system to specific language pairs.

1. Introduction

Hierarchical phrase-based translation (Chiang 2005) is one of the current promising approaches to statistical machine translation (SMT). Hiero SMT systems are based on probabilistic synchronous context-free grammars (SCFGs) whose translation rules

* University of Cambridge, Department of Engineering. CB2 1PZ Cambridge, U.K.
E-mail: {ad465,gwb24,wjb31}@eng.cam.ac.uk.
** University of Vigo, Department of Signal Processing and Communications, 36310 Vigo, Spain.
E-mail: {giglesia,erbanga}@gts.tsc.uvigo.es.

Submission received: 30 October 2009; revised submission received: 24 February 2010; accepted for publication: 10 April 2010.

can be extracted automatically from word-aligned parallel text. These grammars can produce a very rich space of candidate translations and, relative to simpler phrase-based systems (Koehn, Och, and Marcu 2003), the power of Hiero is most evident in translation between dissimilar languages, such as English and Chinese (Chiang 2005, 2007). Hiero is able to learn and apply complex patterns in movement and translation that are not possible with simpler systems. Hiero can also be used to good effect on "simpler" problems, such as translation between English and Spanish (Iglesias et al. 2009c), even though there is not the same need for the full complexity of movement and translation. If gains in using Hiero are small, however, the computational and modeling complexity involved are difficult to justify. Such concerns would vanish if there were reliable methods to match Hiero complexity for specific translation problems. Loosely put, it would be a good thing if the complexity of a system was somehow proportional to the improvement in translation quality the system delivers.

Another notable current trend in SMT is system combination. Minimum Bayes Risk decoding is widely used to rescore and improve hypotheses produced by individual systems (Kumar and Byrne 2004; Tromble et al. 2008; de Gispert et al. 2009), and more aggressive system combination techniques which synthesize entirely new hypotheses from those of contributing systems can give even greater translation improvements (Rosti et al. 2007; Sim et al. 2007). It is now commonplace to note that even the best available individual SMT system can be significantly improved upon by such techniques. This puts a burden on the underlying SMT systems which is somewhat unusual in NLP. The requirement is not merely to produce a single hypothesis that is as good as possible. Ideally, the SMT systems should generate large collections of candidate hypotheses that are simultaneously diverse and of good quality.

Relative to these concerns, previously published descriptions of Hiero have noted certain limitations. **Spurious ambiguity** (Chiang 2005) was described as

> a situation where the decoder produces many derivations that are distinct yet have the same model feature vectors and give the same translation. This can result in *n*-best lists with very few different translations which is problematic for the minimum-error-rate training algorithm ...

This is due in part to the **cube pruning** procedure (Chiang 2007), which enumerates all distinct hypotheses to a fixed depth by means of *k*-best hypothesis lists. If enumeration was not necessary, or if the lists could be arbitrarily deep, there might still be many duplicate derivations, but at least the hypothesis space would not be impoverished.

Spurious ambiguity is also related to **overgeneration** (Varile and Lau 1988; Wu 1997; Setiawan et al. 2009). For our purposes we say that overgeneration occurs when different derivations based on the same set of rules give rise to different translations. An example is given in Figure 1.

This process is not necessarily a bad thing in that it allows new translations to be synthesized from rules extracted from training data; a strong target language model, such as a high order *n*-gram, is typically relied upon to discard unsuitable hypotheses. Overgeneration does complicate translation, however, in that many hypotheses are introduced only to be subsequently discarded. The situation is further complicated by **search errors**. Any search procedure which relies on pruning during search is at risk of search errors and the risk is made worse if the grammars tend to introduce many similar scoring hypotheses. In particular we have found that cube pruning is very prone to search errors, that is, the hypotheses produced by cube pruning are not the top scoring hypotheses which should be found under the Hiero grammar (Iglesias et al. 2009b).

R^1: $X \rightarrow \langle s_1\ X, A\ X \rangle$
R^2: $X \rightarrow \langle X\ s_3, C\ X \rangle$
R^3: $X \rightarrow \langle s_2, B \rangle$

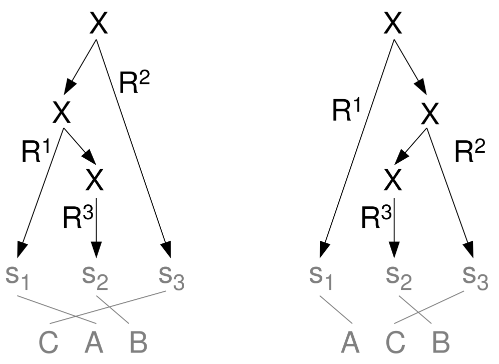


Figure 1
Example of multiple translation sequences from a simple grammar fragment showing variability in reordering in translation of the source sequence *abc*.

These limitations are clearly related to each other. Moreover, they become more problematic as the amount of parallel text grows. As the number of rules in the grammar increases, the grammars become more expressive, but the ability to search them does not improve. This leads to a widening gap between the expressive power of the grammar and the ability to search it to find good and diverse hypotheses.

In this article we describe the following two refinements to Hiero which are intended to address some of the limitations in its original formulation.

Lattice-based hierarchical translation We describe how the cube pruning procedure can be replaced by standard operations with Weighted Finite State Transducers (WFSTs) so that Hiero uses translation lattices rather than *n*-best lists in search. We find that keeping partial translation hypotheses in lattice form greatly reduces search errors. In some instances it is possible to perform translation without any pruning at all so that search errors are completely eliminated. Consistent with the observation by Chiang (2005), this leads to improvements in minimum error rate training. Furthermore, the direct generation of translation lattices can improve gains from subsequent language model and Minimum Bayes Risk (MBR) rescoring.

Shallow-*n* grammars and additional nonterminal categories Nonterminals can be incorporated into hierarchical translation rules for the purpose of tuning the size of the Hiero search space for individual language pairs. Shallow-*n* grammars are described and shown to control the level of rule nesting, low-level rule catenation, and the minimum and maximum spans of individual translation rules. In translation experiments we find that a shallow-1 grammar (one level of rule nesting) is sufficiently expressive for Arabic-to-English translation, but that a shallow-3 grammar is required in Chinese-to-English translation to match the performance of a full Hiero system that allows arbitrary rule nesting. These nonterminals are introduced to control the Hiero search space and do not require estimation from annotated—or parsed—parallel text, as can be required by translation systems based on linguistically motivated grammars. We use this approach as the basis of a general approach to SMT modeling. To control overgeneration, we revisit the synchronous context-free grammar defined by hierarchical rules and take a shallow-1 grammar as a starting point. We then increase the complexity of the rules until the desired translation quality is found.

With these refinements we find that hierarchical phrase-based translation can be efficiently carried out with no (or minimal) search errors in large-data tasks and can achieve state-of-the-art translation performance.

There are many benefits to formulating Hiero translation in terms of WFSTs. Following the manner in which Knight and Al-Onaizan (1998), Kumar, Deng, and Byrne (2006), and Graehl, Knight, and May (2008) elucidate other machine translation models, we can use WFST operations to make the operations of the Hiero decoder very clear. The simplicity of the analysis makes it possible to focus on the underlying grammars and avoid the complexities of heuristic search procedures. Once the decoder is formulated, implementation is mostly straightforward using standard WFST techniques developed for language processing (Mohri, Pereira, and Riley 2002). What difficulties arise are due to using finite state techniques with grammars which are not themselves finite state. We will show, however, that the basic operations which need to be performed, such as extracting sufficient statistics for minimum error rate training, can be done relatively easily and naturally.

1.1 Overview

In Section 2 we describe HiFST, which is a hierarchical phrase-based translation system based on the OpenFST WFST libraries (Allauzen et al. 2007). We describe how translation lattices can be generated over the Cocke-Younger-Kasami (CYK) grid used for parsing under Hiero. We also review some modeling issues needed for practical translation, such as the efficient handling of source language deletions and the extraction of statistics for minimum error rate training. This requires running HiFST in "alignment mode" (Section 2.3) to find all the rule derivations that generate a given set of translation hypotheses.

In Section 3 we investigate parameters that control the size and nature of the hierarchical phrase-based search space as defined by hierarchical translation rules. To efficiently explore the largest possible space and avoid pruning in search, we introduce ways to easily adapt the grammar to the reordering needs of each language pair. We describe the use of additional nonterminal categories to limit the degree of rule nesting, and can directly control the minimum or maximum span each translation rule can cover.

In Section 4 we report detailed translation results for Arabic-to-English and Chinese-to-English, and review translation results for Spanish-to-English and Finnish-to-English translation. In these experiments we contrast the performance of lattice-based and cube pruning hierarchical decoding and we measure the impact on processing time and translation performance due to changes in search parameters and grammar configurations. We demonstrate that it is easy and feasible to compute the marginal instead of the Viterbi probabilities when using WFSTs, and that this yields gains in translation performance. And finally, we show that lattice-based translation performs significantly better than *k*-best lists for the task of combining translation hypotheses generated from alternative morphological segmentations of the data via lattice-based MBR decoding.

2. Hierarchical Translation and Alignment with WFSTs

Hierarchical phrase-based rules define a synchronous context-free grammar (CFG) and a particular search space of translation candidates. Table 1 shows the type of rules included in a standard hierarchical phrase-based grammar, where **T** denotes the terminals (words) and $\sim$ is a bijective function that relates the source and target nonterminals of

Table 1
Rules contained in the standard hierarchical grammar.

standard hierarchical grammar	
$S \rightarrow \langle X,X \rangle$	glue rule 1
$S \rightarrow \langle S\ X,S\ X \rangle$	glue rule 2
$X \rightarrow \langle \gamma,\alpha,\sim \rangle\ ,\ \gamma,\alpha \in \{X \cup \mathbf{T}\}^+$	hiero rules

each rule (Chiang 2007). This function is defined if there are at least two nonterminals, and for clarity of presentation may be omitted in general rule discussions. When $\gamma, \alpha \in \{\mathbf{T}\}^+$, that is, the rule contains no nonterminals, the rule is a simple lexical phrase pair.

The HiFST translation system is based on a variant of the CYK algorithm closely related to CYK+ (Chappelier and Rajman 1998). Parsing follows the description of Chiang (2005, 2007); it maintains back-pointers and employs hypothesis recombination without pruning. The underlying model is a probabilisitic synchronous CFG consisting of a set $\mathbf{R} = \{R^r\}$ of rules $R^r : N^r \rightarrow \langle \gamma^r,\alpha^r \rangle\ /\ p^r$, with "glue" rules, $S \rightarrow \langle X,X \rangle$ and $S \rightarrow \langle S\ X,S\ X \rangle$. $\mathbf{N}$ denotes the set of nonterminal categories (examples are given in Section 3), and p^r denotes the rule probability, typically transformed to a cost c^r; unless otherwise noted we use the tropical semiring, so $c^r = -\log p^r$. $\mathbf{T}$ denotes the terminals (words), and the grammar builds parses based on strings $\gamma, \alpha \in \{\mathbf{N} \cup \mathbf{T}\}^+$. Each cell in the CYK grid is specified by a nonterminal 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 CFG 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 back-pointers 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.

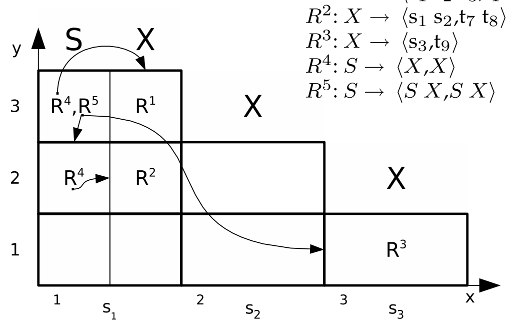
The ultimate objective is the word lattice $\mathcal{L}(S, 1, J)$ which corresponds to all the analyses that cover the source sentence s_1^J. Once this is built, we can apply a target language model to $\mathcal{L}(S, 1, J)$ to obtain the final target language translation lattice (Allauzen, Mohri, and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

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 $R(N, x, y)$, that is, for $r \in R(N, x, y)$, the rule $N \rightarrow \langle \gamma^r,\alpha^r \rangle$ 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 corresponding to the elements of $\alpha^r = \alpha_1^r ... \alpha_{|\alpha^r|}^r$. If an α_i^r is a terminal, creating its lattice is straightforward. If α_i^r is a nonterminal, it refers to a cell (N', x', y') lower in the grid identified by the back-pointer $BP(N, x, y, r, i)$; in this case, the lattice used is $\mathcal{L}(N', x', y')$. Taken together,

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



R^1: $X \rightarrow \langle \mathrm{s}_1\ \mathrm{s}_2\ \mathrm{s}_3, \mathrm{t}_1\ \mathrm{t}_2 \rangle$
R^2: $X \rightarrow \langle \mathrm{s}_1\ \mathrm{s}_2, \mathrm{t}_7\ \mathrm{t}_8 \rangle$
R^3: $X \rightarrow \langle \mathrm{s}_3, \mathrm{t}_9 \rangle$
R^4: $S \rightarrow \langle X, X \rangle$
R^5: $S \rightarrow \langle S\ X, S\ X \rangle$

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

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

Figure 2
Production of target lattice $\mathcal{L}(S,1,3)$ using translation rules within CYK grid for sentence $s_1 s_2 s_3$. The grid is represented here in two dimensions (x, y). In practice only the first column accepts both nonterminals (S, X). For this reason it is divided into two subcolumns.

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

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 $R(N,x,y)$:

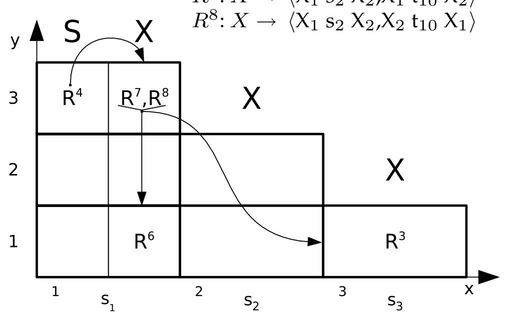
$$\mathcal{L}(N,x,y) = \bigoplus_{r \in R(N,x,y)} \mathcal{L}(N,x,y,r) \tag{3}$$

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 $L(N,x,y,r)$ prior to the operation of Equation (3).

2.1.1 An Example of Phrase-based Translation. Figure 2 illustrates this process for a three-word source sentence $s_1 s_2 s_3$ under monotonic 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 three standard phrases, that is, rules with only terminals (R^1, R^2, R^3), and the glue rules (R^4, R^5). Arrows represent back-pointers to lower-level cells. We are interested in the uppermost 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 back-pointers of these two rules. This process is explicitly derived in the right-hand side of Figure 2.

2.1.2 An Example of Hierarchical Translation. Figure 3 shows a hierarchical scenario for the same sentence. Three rules, R^6, R^7, R^8, are added to the example of Figure 2, 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 2; these are shown within $\{\}$.

R^6: $X \rightarrow \langle \mathrm{s_1}, \mathrm{t_{20}} \rangle$
R^7: $X \rightarrow \langle \mathrm{X_1\ s_2\ X_2}, \mathrm{X_1\ t_{10}\ X_2} \rangle$
R^8: $X \rightarrow \langle \mathrm{X_1\ s_2\ X_2}, \mathrm{X_2\ t_{10}\ X_1} \rangle$



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

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

Figure 3
Translation as in Figure 2 but with additional rules R^6,R^7,R^8. Lattices previously derived appear within {}.

2.2 A Procedure for Lattice Construction

Figure 4 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)–(3). For every rule, each element of the target side (lines 3,4) is checked as terminal or nonterminal (Equation (2)). If it is a terminal element (line 5), a simple acceptor is built. If it is a nonterminal (line 6), the lattice associated to its back-pointer 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 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).

We now address several important aspects of efficient implementation.

```
 1: function buildFst(N,x,y)
 2:     if $\exists\ \mathcal{L}(N,x,y)$ then return $\mathcal{L}(N,x,y)$
 3:     for $r \in R(N,x,y)$, $R^r : N \rightarrow \langle\gamma,\alpha\rangle$ do
 4:         for $i = 1...|\alpha|$ do
 5:             if $\alpha_i \in \mathbf{T}$ then $\mathcal{L}(N,x,y,r,i) = \mathcal{A}(\alpha_i)$
 6:             else
 7:                 $(N',x',y') = BP(\alpha_i)$
 8:                 $\mathcal{L}(N,x,y,r,i) = \text{buildFst}(N',x',y')$
 9:         $\mathcal{L}(N,x,y,r) = \bigotimes_{i=1..|\alpha|} \mathcal{L}(N,x,y,r,i)$
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 4
Recursive lattice construction from a CYK grid.

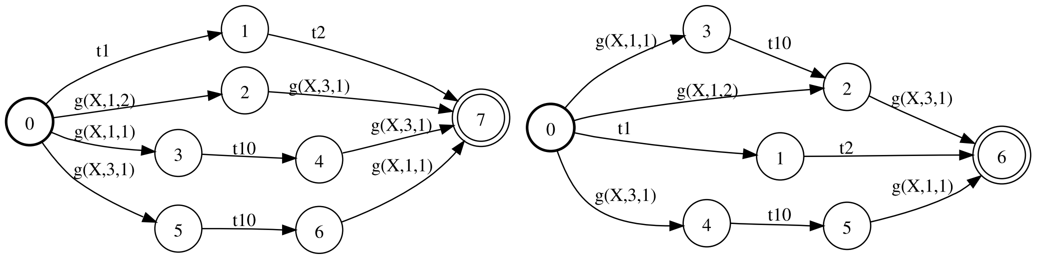


Figure 5
Delayed translation WFST with derivations from Figures 2 and 3 before (left) and after minimization (right).

2.2.1 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 for 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 back-pointer $(N', x', y') = BP(N, x, y, r, i)$, these special arcs are introduced as

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

The resulting lattices $\mathcal{L}(N, x, y)$ are a mix of target language words and lattice pointers (Figure 5, left). Each still represents the entire search space of all translation hypotheses covering the span, however. Importantly, operations on these lattices—such as lossless size reduction via determinization and minimization (Mohri, Pereira, and Riley 2002)—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 5 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. Not all redundancy is removed, however, because 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, Pereira, and Riley (2000), closely related to Recursive Transition Networks (Woods 1970; Mohri 1997). Its implementation uses the infrastructure provided by the OpenFST libraries for delayed composition.

2.2.2 Top-level Pruning and Search Pruning. The final translation lattice $\mathcal{L}(S, 1, J)$ can be quite large after the pointer arcs are expanded. We therefore apply a word-based

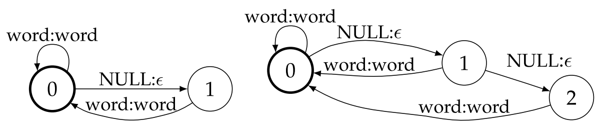


Figure 6
Transducers for filtering up to one (left) or two (right) consecutive deletions.

language model via WFST composition (Allauzen et al. 2007) and perform likelihood-based pruning based on the combined translation and language model scores. We call this top-level pruning because it is performed over the topmost lattice.

Pruning can also be performed on the sublattices in each cell 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. These pruning strategies can be very selective, for example allowing the pruning threshold to depend on the height of the cell in the grid. In this way the risk of search errors can be controlled.

The same *n*-gram language model can be used for top-level pruning and search pruning, although different WFST realizations are required. For top-level pruning, a standard implementation as described by Allauzen et al. (2007) is appropriate. For search pruning, the WFST must allow for incomplete language model histories, because many sublattice paths are incomplete translation hypotheses which do not begin with a sentence-start marker. The language model acceptor is constructed so that initial substrings of length less than the language model order are assigned no weight under the language model.

2.2.3 Constrained Source Word Deletion. As a practical matter it can be useful to allow SMT systems to delete some source words rather than to enforce their translation. Deletions can be allowed in Hiero by including in the grammar a set of special deletion rules of the type: $X \rightarrow \langle s, \mathrm{NULL} \rangle$. Unconstrained application of these rules can lead to overly large and complex search spaces, however. We therefore limit the number of consecutive source word deletions as we explore each cell of the CYK grid. This is done by standard composition with an unweighted transducer that maps any word to itself, and up to k NULL tokens to ϵ arcs. In Figure 6 this simple transducer for $k = 1$ and $k = 2$ is drawn. Composition of the lattice in each cell with this transducer filters out all translations with more than k consecutive deleted words.

2.3 Hierarchical Phrase-Based Alignment with WFSTs

We now describe a method to apply our decoder in alignment mode. The objective in alignment is to recover all the derivations which can produce a given translation. We do this rather than keep track of the rules used during translation, because we find it faster and more efficient first to generate translations and then, by running the system as an aligner with a constrained target space, to extract all the relevant derivations with their costs. As will be discussed in Section 2.3.1, this is useful for minimum error training, where the contribution of each feature to the overall hypothesis cost is required for system optimization.

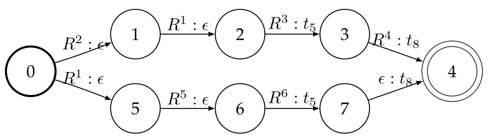


Figure 7
Transducer encoding simultaneously rule derivations $R^2R^1R^3R^4$ and $R^1R^5R^6$, and the translation t_5t_8. The input sentence is $s_1s_2s_3$ and the grammar considered here contains the following rules: R^1: $S\rightarrow\langle X,X\rangle$, R^2: $S\rightarrow\langle S\ X,S\ X\rangle$, R^3: $X\rightarrow\langle \mathrm{s}_1,\mathrm{t}_5\rangle$, R^4: $X\rightarrow\langle \mathrm{s}_2\ \mathrm{s}_3,\mathrm{t}_8\rangle$, R^5: $X\rightarrow\langle \mathrm{s}_1\ X\ \mathrm{s}_3,X\ \mathrm{t}_8\rangle$ and R^6: $X\rightarrow\langle \mathrm{s}_2,\mathrm{t}_5\rangle$.

Conceptually, we would like to create a transducer that represents the mapping from all possible rule derivations to all possible translations, and then compose this transducer with an acceptor for the translations of interest. Creating this transducer which maps derivations to translations is not feasible for large translation grammars, so we instead keep track of rules as they are used to generate a particular translation output. We introduce two modifications into lattice construction over the CYK grid described in Section 2.2:

1. In each cell transducers are constructed which map rule sequences to the target language translation sequences they produce. In each transducer the output strings are all possible translations of the source sentence span covered by that cell; the input strings are all the rule derivations that generate those translations. The rule derivations are expressed as sequences of rule indices r given the set of rules $\mathbf{R} = \{R^r\}$.
2. As these transducers are built they are composed with acceptors for subsequences of the reference translations so that any translations not present in the given set of reference translations are removed. In effect, this replaces the general target language model used in translation with an unweighted acceptor which accepts only specific sentences.

For alignment, Equations (1) and (2) are redefined as

$$\mathcal{L}(N,x,y,r) = A_T(r,\epsilon) \bigotimes_{i=1..|\alpha^r|} \mathcal{L}(N,x,y,r,i) \tag{5}$$

$$\mathcal{L}(N,x,y,r,i) = \begin{cases} \mathcal{A}_T(\epsilon,\alpha_i) \text{ if } \alpha_i \in \mathbf{T} \\ \mathcal{L}(N',x',y') \text{ otherwise} \end{cases} \tag{6}$$

where $\mathcal{A}_T(r,t)$, $R^r \in \mathbf{R}$, $t \in \mathbf{T}$ returns a single-arc transducer which accepts the symbol r in the input language (rule indices) and the symbol t in the output language (target words). The weight assigned to each arc is the same in alignment as in translation. With these definitions the goal lattice $\mathcal{L}(S,1,J)$ is now a transducer with rule indices in the input symbols and target words in the output symbols. A simple example is given in Figure 7 where two rule derivations for the translation t_5t_8 are represented by the transducer.

As we are only interested in those rule derivations that generate the given target references, we can discard non-desired translations via standard FST composition of

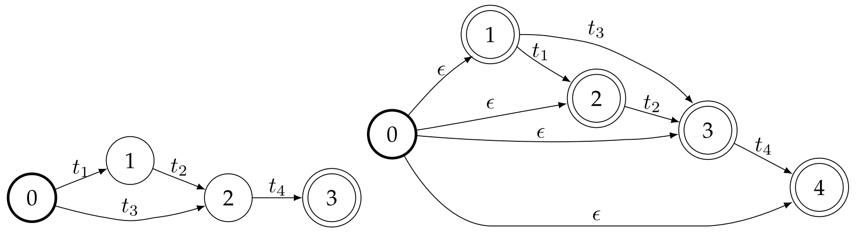


Figure 8
Construction of a substring acceptor. An acceptor for the strings $t_1t_2t_4$ and t_3t_4 (left) and its substring acceptor (right). In alignment the substring acceptor can be used to filter out undesired partial translations via standard FST composition operations.

the lattice transducer with the given reference acceptor. In principle, this would be done in the uppermost cell of the CYK, once the complete source sentence has been covered. However, keeping track of all possible rule derivations and all possible translations until the last cell may not be computationally feasible for many sentences. It is more desirable to carry out this filtering composition in lower-level cells while constructing the lattice over the CYK grid so as to avoid storing an increasing number of undesired translations and derivations in the lattice. The lattice in each cell should contain translations formed only from substrings of the references.

To achieve this we build an unweighted substring acceptor that accepts all substrings of each target reference string. For instance, given the reference string $t_1t_2 \ldots t_J$, we build an acceptor for all substrings $t_i \ldots t_j$, where $1 \leq i \leq j \leq J$. This is applied to lattices in all cells (x, y) that do not span the whole sentence. In the uppermost cell we compose with the reference acceptor which accepts only complete reference strings. Given a lattice of target references, the unweighted substring acceptor is built as:

1. change all non-initial states into final states
2. add one initial state and add ϵ arcs from it to all other states

Figure 8 shows an example of a substring acceptor for the two references $t_1t_2t_4$ and t_3t_4. The substring acceptor also accepts an empty string, accounting for those rules that delete source words, which in other words translate into NULL. In some instances the final composition with the reference acceptor might return an empty lattice. If this happens there is no rule sequence in the grammar that can generate the given source and target sentences simultaneously.

We have described the use of transducers to encode mappings from rule derivations to translations. These transducers are somewhat impoverished relative to parse trees and parse forests, which are more commonly used to encode this relationship. It is easy to map from a parse tree to one of these transducers but the reverse essentially requires re-parsing to recreate the tree structure. The structures of the parse trees associated with a translation are not needed by many algorithms, however. In particular, parameter optimization by MERT requires only the rules involved in translation. Our approach keeps only what is needed by such algorithms. This approach also has practical advantages such as the ability to align directly to k-best lists or lattices of reference translations.

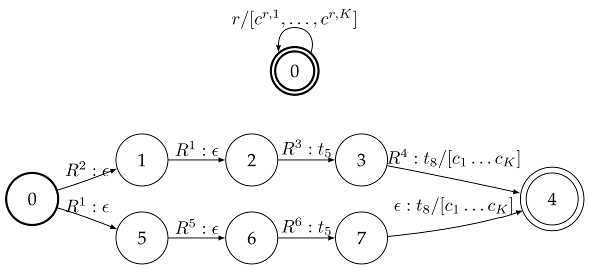

$r/[c^{r,1},\ldots,c^{r,K}]$
0
$R^2:\epsilon$
$R^1:\epsilon$
$R^1:\epsilon$
$R^3:t_5$
$R^4:t_8/[c_1\ldots c_K]$
$\epsilon:t_8/[c_1\ldots c_K]$
$R^5:\epsilon$
$R^6:t_5$

Figure 9
One arc from a rule acceptor that assigns a vector of K feature weights to each rule (top) and the result of composition with the transducer of Figure 7 (after weight-pushing) (bottom). The components of the final K-dimensional weight vector agree with the feature weights of the rule sequence, for example $c_k = c^{2,k} + c^{1,k} + c^{3,k} + c^{4,k}$ for $k = 1 \ldots K$.

2.3.1 Extracting Feature Values from Alignments. As described by Chiang (2007), scores are associated with hierarchical translation rules through a factoring into features within a log-linear model (Och and Ney 2002). We assume that we have a collection of K features and that the cost c^r of each rule R^r is $c^r = \sum_{k=1}^{K} \lambda_k c^{r,k}$, where $c^{r,k}$ is the value of the k^{th} feature value for the r^{th} rule and λ_k is the weight assigned to the k^{th} feature for all rules. For a parse which makes use of the rules $R^{r_1} \ldots R^{r_N}$, its cost $\sum_{n=1}^{N} c^{r_n}$ can therefore be written as $\sum_{k=1}^{K} \lambda_k \sum_{n=1}^{N} c^{r_n,k}$. The quantity $\sum_{n=1}^{N} c^{r_n,k}$ is the contribution by the k^{th} feature to the overall translation score for that parse. These are the quantities which need to be extracted from alignment lattices for use in procedures such as minimum error rate training for estimation of the feature weights λ_k.

The procedure described in Section 2.3 produces alignment lattices with scores consistent with the total parse score. Further steps must be taken to factor this overall score to identify the contribution due to individual features or translation rules. We introduce a rule acceptor which accepts sequences of rule indices, such as the input sequences of the alignment transducer, and assigns weights in the form of K-dimensional vectors. Each component of the weight vector corresponds to the feature value for that rule. Arcs have the form $0 \xrightarrow{R^r/w^r} 0$ where $w^r = [c^{r,1}, \ldots, c^{r,K}]$. An example of composition with this rule acceptor is given in Figure 9 to illustrate how feature scores are mapped to components of the weight vector. The same operations can be applied to the (unweighted) alignment transducer on a much larger scale to extract the statistics needed for minimum error rate training.

We typically apply this procedure in the tropical semiring (Viterbi likelihoods), so that only the best rule derivation that generated each translation candidate is taken into account when extracting feature contributions for MERT. However, given the alignment transducer $\mathcal{L}$, this could also be performed in the log semiring (marginal likelihoods), taking into account the feature contributions from all rule derivations, for each translation candidate. This would be adequate if the translation system also carried out decoding in the log semiring, an experiment which is partially explored in Section 4.4.

We note that the contribution of the language model to the overall translation score cannot be calculated in this scheme, since the language model score cannot be factored in terms of rules. To obtain the language model contribution, we simply carry out WFST composition (Allauzen et al. 2007) between an unweighted acceptor of the target

R^1: $S \to \langle X, X \rangle$
R^2: $S \to \langle S\ X, S\ X \rangle$
R^3: $X \to \langle X\ \mathrm{s}_3, \mathrm{t}_5\ X \rangle$
R^4: $X \to \langle X\ \mathrm{s}_4, \mathrm{t}_3\ X \rangle$
R^5: $X \to \langle \mathrm{s}_1\ \mathrm{s}_2, \mathrm{t}_1\ \mathrm{t}_2 \rangle$
R^6: $X \to \langle \mathrm{s}_4, \mathrm{t}_7 \rangle$

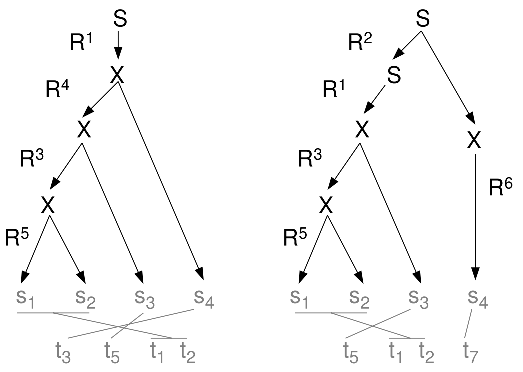


Figure 10
Hierarchical translation grammar example and two parse trees with different levels of rule nesting for the input sentence $s_1s_2s_3s_4$.

sentences and the n-gram language model used in translation. After determinization, the cost of each path in the acceptor is then the desired LM feature contribution.

3. Shallow-*n* Translation Grammars: Translation Search Space Refinements

In this section we describe shallow-*n* grammars in order to reduce Hiero overgeneration and adapt the grammar complexity to specific language pairs; the ultimate goal is to define the most constrained grammar that is capable of generating the desired movement and translation, so that decoding can be performed without search errors.

Hiero can provide varying degrees of complexity in movement and translation. Consider the example shown in Figure 10, which shows a hierarchical grammar defined by six rules. For the input sentence $s_1s_2s_3s_4$, there are two possible parse trees as shown; the rule derivations for each tree are $R^1R^4R^3R^5$ and $R^2R^1R^3R^5R^6$. Along with each tree is shown the translation generated and the phrase-level alignment. Comparing the two trees and alignments, the leftmost tree makes use of more reordering when translating from source to target through the nested application of the hierarchical rules R^3 and R^4. For some language pairs this level of reordering may be required in translation, but for other language pairs it may lead to overgeneration of unwanted hypotheses. Suppose the grammar in this example is modified as follows:

1. A nonterminal X^0 is introduced into hierarchical translation rules

R^3:$X \to \langle X^0\ \mathrm{s}_3, \mathrm{t}_5\ X^0 \rangle$
R^4:$X \to \langle X^0\ \mathrm{s}_4, \mathrm{t}_3\ X^0 \rangle$

2. Rules for lexical phrases are applied to X^0

R^5:$X^0 \to \langle \mathrm{s}_1\ \mathrm{s}_2, \mathrm{t}_1\ \mathrm{t}_2 \rangle$
R^6:$X^0 \to \langle \mathrm{s}_4, \mathrm{t}_7 \rangle$

These modifications exclude parses in which hierarchical translation rules generate other hierarchical rules, except at the 0^{th} level which generate lexical phrases. Consequently the left most tree of Figure 10 cannot be generated and $t_5t_1t_2t_7$ is the only allowable translation of $s_1s_2s_3s_4$. We call this a 'shallow-1' grammar: The maximum degree of rule nesting allowed is 1 and only the glue rule can be applied above this level.

The use of additional nonterminal categories is an elegant way to easily control important aspects that can have a strong impact on the search space. A shallow-n translation grammar can be formally defined as:

1. the usual nonterminal S
2. a set of nonterminals $\{X^0, \ldots, X^N\}$
3. two glue rules: $S \rightarrow \langle X^N, X^N \rangle$ and $S \rightarrow \langle S\ X^N, S\ X^N \rangle$
4. hierarchical translation rules for levels $n = 1, \ldots, N$:
   R: $X^n \rightarrow \langle \gamma, \alpha, \sim \rangle$, $\gamma, \alpha \in \{\{X^{n-1}\} \cup \mathbf{T}\}^+$
   with the requirement that α and γ contain at least one X^{n-1}
5. translation rules which generate lexical phrases:
   R: $X^0 \rightarrow \langle \gamma, \alpha \rangle$, $\gamma, \alpha \in \mathbf{T}^+$

3.1 Avoiding Some Spurious Ambiguity

The added requirement in condition (4) in the definition of shallow-n grammars is included to avoid some instances in which multiple parses lead to the same translation. It is not absolutely necessary but it can be added without any loss in representational capability. To see the effect of this constraint, consider the following example with a source sentence $s_1\ s_2$ and a shallow-1 grammar defined by these four rules:

R^1: $S \rightarrow \langle X^1, X^1 \rangle$
R^2: $X^1 \rightarrow \langle s_1\ s_2, t_2\ t_1 \rangle$
R^3: $X^1 \rightarrow \langle s_1\ X^0, X^0\ t_1 \rangle$
R^4: $X^0 \rightarrow \langle s_2, t_2 \rangle$

There are two derivations $\mathrm{R}^1\mathrm{R}^2$ and $\mathrm{R}^1\mathrm{R}^3\mathrm{R}^4$ which yield identical translations. However R^2 would not be allowed under the constraint introduced here because it does not rewrite an X^1 to an X^0.

3.2 Structured Long-Distance Movement

The basic formulation of shallow-n grammars allows only the upper-level nonterminal category S to act within the glue rule. This can prevent some useful long-distance movement, as might be needed to translate Arabic sentences in Verb-Subject-Object order into English. It often happens that the initial Arabic verb requires long-distance movement, but the subject which follows can be translated in monotonic order. For instance, consider the following Romanized Arabic sentence:

TAlb	AlwzrA'	AlmjtmEyn	Alywm	fy dm$q	<lY	...
(CALLED)	(the ministers)	(gathered)	(today)	(in Damascus)	(FOR)	...

where the verb 'TAlb' must be translated into English so that it follows the translations of the five subsequent Arabic words 'AlwzrA' AlmjtmEyn Alywm fy dm$q', which are themselves translated monotonically. A shallow-1 grammar cannot generate this movement except in the relatively unlikely case that the five words following the verb can be translated as a single phrase.

A more powerful approach is to define grammars which allow low-level rules to form movable groups of phrases. Additional nonterminals $\{M^k\}$ are introduced to allow successive generation of k nonterminals X^{N-1} in monotonic order for both languages, where $K_1 \leq k \leq K_2$. These act in the same manner as the glue rule does in the uppermost level. Applying M^k nonterminals at the N–1 level allows one hierarchical rule to perform a long-distance movement over the tree headed by M^k.

We further refine the definition of shallow-*n* grammars by specifying the allowable values of k for the successive productions of nonterminals X^{N-1}. There are many possible ways to formulate and constrain these grammars. If $K_2 = 1$, then the grammar is equivalent to the previous definition of shallow-*n* grammars, because monotonic production is only allowed by the glue rule of level N. If $K_1 = 1$ and $K_2 > 1$, then the search space defined by the grammar is greater than the standard shallow-*n* grammar as it includes structured long-distance movement. Finally, if $K_1 > 1$ then the search space is different from standard shallow-*n* as the *n* level is only used for long-distance movement.

Introduction of M^k nonterminals redefines shallow-*n* grammars as:

1. the usual nonterminal S
2. a set of nonterminals $\{X^0, \ldots, X^N\}$
3. a set of nonterminals $\{M^{K_1}, \ldots, M^{K_2}\}$ for $K_1 = 1, 2;\ K_1 \leq K_2$
4. two glue rules: $S \rightarrow \langle X^N, X^N \rangle$ and $S \rightarrow \langle S\ X^N, S\ X^N \rangle$
5. hierarchical translation rules for level N:
   R: $X^N \rightarrow \langle \gamma, \alpha, \sim \rangle\ , \gamma, \alpha \in \{\{M^{K_1}, \ldots, M^{K_2}\} \cup \mathbf{T}\}^+$
   with the requirement that α and γ contain at least one M^k
6. hierarchical translation rules for levels $n = 1, \ldots, N-1$:
   R: $X^n \rightarrow \langle \gamma, \alpha, \sim \rangle\ , \gamma, \alpha \in \{\{X^{n-1}\} \cup \mathbf{T}\}^+$
   with the requirement that α and γ contain at least one X^{n-1}
7. translation rules which generate lexical phrases:
   R: $X^0 \rightarrow \langle \gamma, \alpha \rangle\ , \gamma, \alpha \in \mathbf{T}^+$
8. rules which generate k nonterminals X^{N-1}:
   if $K_1 == 2$:
   R: $M^k \rightarrow \langle X^{N-1}\ M^{k-1}, X^{N-1}\ M^{k-1} \rangle$, for $k = 3, \ldots, K_2$
   R: $M^2 \rightarrow \langle X^{N-1}\ X^{N-1}, X^{N-1}\ X^{N-1}, I \rangle$
   if $K_1 == 1$:
   R: $M^k \rightarrow \langle X^{N-1}\ M^{k-1}, X^{N-1}\ M^{k-1} \rangle$, for $k = 2, \ldots, K_2$
   R: $M^1 \rightarrow \langle X^{N-1}, X^{N-1} \rangle$

where I denotes the identity function that enforces monotonocity in the nonterminals. For example, with a shallow-1 grammar, M^3 leads to the monotonic production of three nonterminals X^0, which leads to the production of three lexical phrase pairs; these can be moved with a hierarchical rule of level 1. This is graphically represented by the leftmost tree in Figure 11. With a shallow-2 grammar, M^2 leads to the monotonic production of

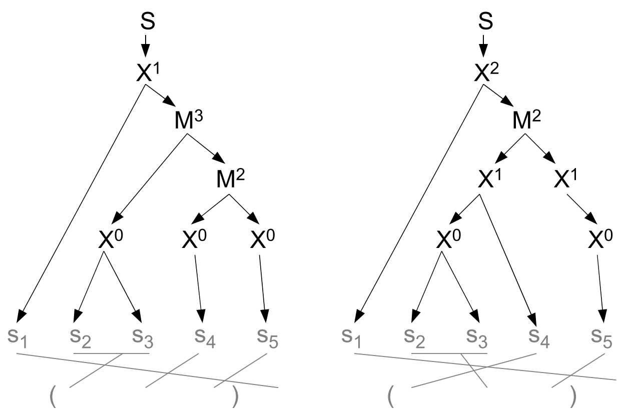


Figure 11
Movement allowed by two grammars: shallow-1, with $K_1 = 1$, $K_2 = 3$ (left), and shallow-2, with $K_1 = 1$, $K_2 = 3$ (right). Both grammars allow movement of the bracketed term as a unit. Shallow-1 requires that translation within the object moved be monotonic whereas shallow-2 allows up to two levels of reordering.

two nonterminals X^1, a movement represented by the rightmost tree in Figure 11. This movement cannot be achieved with a shallow-1 grammar.

3.3 Minimum and Maximum Rule Span

It is useful to define two parameters which further control the application of hierarchical translation rules in generating the search space. Parameters *hmax* and *hmin* specify the maximum and minimum height at which any hierarchical translation rule can be applied in the CYK grid. In other words, a hierarchical rule can only be applied in cell (x, y) if $hmin \leq y \leq hmax$. Note that these parameters can also be set independently for each nonterminal category.

3.4 Verb Movement Grammars for Arabic-to-English Translation

Following the discussion which motivated this section, we wish to model movement of Arabic verbs when translating into English. We add to the shallow-*n* grammars a **verb restriction** so that the hierarchical translation rules (5) apply only if the source language string γ contains a verb. This encourages translations in which the Arabic verb is moved at the uppermost level N.

3.5 Grammars Used for SMT Experiments

We now define the hierarchical grammars for the translation experiments which we describe next.

Shallow-1,2,3 : Shallow-*n* grammars for $n = 1, 2, 3$. These grammars do not incorporate any monotonicity constraints, that is $K_1 = K_2 = 1$.

Shallow-1, $\mathbf{K_1 = 1}$, $\mathbf{K_2 = 3}$: hierarchical rules with one nonterminal can reorder a monotonic production of up to three target language phrases of level 0.

Shallow-1, $K_1 = 1, K_2 = 3$, vo : hierarchical rules with one nonterminal can reorder a monotonic catenation of up to three target language phrases of level 0, but only if one of the source terminals is tagged as a verb.

Shallow-2, $K_1 = 2, K_2 = 3$, vo : two levels of reordering with monotonic production of up to three target language phrases of level 1, but only if one of the source terminals is tagged as a verb.

4. Translation Experiments

In this section we report on hierarchical phrase-based translation experiments with WFSTs. We focus mainly on the NIST Arabic-to-English and Chinese-to-English translation tasks; some results for other language pairs are summarized in Section 4.6. Translation performance is evaluated using the BLEU score (Papineni et al. 2001) as implemented for the NIST 2009 evaluation.[1] The experiments are organized as follows:

- Lattice-based and cube pruning hierarchical decoding (Section 4.2)
- Grammar configurations and search parameters and their effect on translation performance and processing time (Section 4.3)
- Marginalization over translation derivations (Section 4.4)
- Combining translation lattices obtained from alternative morphological decompositions of the input (Section 4.5)

4.1 Experimental Framework

For Arabic-to-English translation we use all allowed parallel corpora in the NIST MT08 (and MT09) Arabic Constrained Data track ($\sim$150M words per language). In addition to reporting results on the MT08 set itself, we make use of 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 a validation set *mt02-05-test*. The *mt02-05-tune* set has 2,075 sentences.

For Chinese-to-English translation we use all available parallel text for the GALE 2008 evaluation;[2] this is approximately 250M words per language. We report translation results on the NIST MT08 set, a development set *tune-nw*, and a validation set *test-nw*. These tuning and test sets contain translations produced by the GALE program and portions of the newswire sections of MT02 through MT05. The *tune-nw* set has 1,755 sentences, and *test-nw* set is similar.

The parallel texts for both language pairs are aligned using MTTK (Deng and Byrne 2008). We extract hierarchical rules from the aligned parallel texts using the constraints developed by Chiang (2007). We further filter the extracted rules by count and pattern as described by Iglesias et al (2009a). The following features are extracted from the parallel data and used to assign scores to translation rules: 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).

1 See `ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v13.pl`
2 See `http://projects.ldc.upenn.edu/gale/data/catalog.html`

We use two types of language model in translation. In first-pass translation we use 4-gram language models estimated over the English side of the parallel text (for each language pair) and a 965 million word subset of monolingual data from the English Gigaword Third Edition (LDC2007T07). These are the language models used if pruning is needed during search. The main language model is a zero-cutoff stupid-backoff (Brants et al. 2007) 5-gram language model, estimated using 6.6B words of English text from the English Gigaword corpus. These language models are converted to WFSTs as needed (Allauzen, Mohri, and Roark 2003); failure transitions are used for correct application of back-off weights. In tuning the systems, standard MERT (Och 2003) iterative parameter estimation under IBM BLEU is performed on the development sets.

4.2 Contrast between HiFST and Cube Pruning

We contrast two hierarchical phrase-based decoders. The first decoder, HCP, is a *k*-best decoder using cube pruning following the description by Chiang (2007); in our implementation, these *k*-best lists contain only unique hypotheses (Iglesias et al. 2009a), which are obtained by extracting the 10,000 best candidates from each cell (including the language model cost), using a priority queue to explore the cross-product of the *k*-best lists from the cells pointed by nonterminals. We find that deeper *k*-best lists (i.e., $k = 100,000$) results in impractical decoding times and that fixed *k*-best list depths yields better performance than use of a likelihood threshold parameter. The second decoder, HiFST, is a lattice-based decoder implemented with WFSTs as described earlier. Hypotheses are generated after determinization under the tropical semiring so that scores assigned to hypotheses arise from a single minimum cost/maximum likelihood derivation.

Translation proceeds as follows. After Hiero translation with optimized feature weights and the first-pass language model, hypotheses are written to disk. For HCP we save translations as 10,000-best lists, whereas HiFST generates word lattices. The first-pass results are then rescored with the main 5-gram language model. In this operation the first-pass language model scores are removed before the main language model scores are applied. We then perform MBR rescoring. For the *n*-best lists we rescore the top 1,000 hypotheses using the negative sentence-level BLEU score as the loss function (Kumar and Byrne 2004); we have found that using a deeper *k*-best list is impractically slow. For the HiFST lattices we use lattice-based MBR search procedures described by Tromble et al. (2008) in an implementation based on standard WFST operations (Allauzen et al. 2007).

4.2.1 Shallow-1 Arabic-to-English Translation. We translate Arabic-to-English with shallow-1 hierarchical decoding: as described in Section 3, nonterminals are allowed only to generate target language phrases. Table 2 shows results for *mt02-05-tune*, *mt02-05-test*, and *mt08*. In this experiment we use MERT to find optimized parameters for HCP and we use these parameter values in HiFST as well. This allows for a close comparison of decoder behavior, independent of parameter optimization.

In these experiments, the first-pass translation quality of the two systems (Table 2 a vs. b) is nearly identical. The most notable difference in the first-pass behavior of the decoders is their memory use. For example, for an input sentence of 105 words, HCP uses 1.2Gb memory whereas HiFST takes only 900Mb under the same conditions. To run HCP successfully requires cube pruning with the first-pass 4-gram language model. By contrast, HiFST requires no pruning during lattice construction and the first pass language model is not applied until the lattice is fully built at the upper most cell of the

Table 2
Contrastive Arabic-to-English translation results (lower-cased IBM BLEU) after first-pass decoding and subsequent rescoring steps. Decoding time reported for *mt02-05-tune* is in seconds per word. Both systems are optimized using MERT over the *k*-best lists generated by HCP.

	decoder	*time*	*mt02-05-tune*	*mt02-05-test*	*mt08*
a	HCP	1.1	52.5	51.9	42.8
	+5g	-	53.4	52.9	43.5
	+5g+MBR	-	53.6	53.0	43.6
b	HiFST	0.5	52.5	51.9	42.8
	+5g	-	53.6	53.2	43.9
	+5g+LMBR	-	54.3	53.7	44.8

CYK grid. For this grammar, HiFST is able to produce exact translations without any search errors.

Search Errors Because both decoders are constrained to use exactly the same features, we can compare 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 HiFST finds a hypothesis with lower cost than HCP. In contrast, HCP never finds any hypothesis with lower cost for any sentence. This is as expected: The HiFST decoder requires no pruning prior to applying the first-pass language model, so search is exact.

Lattice/k-best Quality It is clear from the results that the lattices produced by HiFST yield better rescoring results than the *k*-best lists produced by HCP. This is the case for both 5-gram language model rescoring and MBR. In MT08 rescoring, HCP *k*-best lists yield an improvement of 0.8 BLEU relative to the first-pass baseline, whereas rescoring HiFST lattices yield an improvement of 2.0 BLEU. The advantage of maintaining a large search space in lattice form during decoding is clear. The use of *k*-best lists in HCP limits the gains from subsequent rescoring procedures.

Translation Speed HCP requires an average of 1.1 seconds per input word. HiFST 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 independently processing each distinct hypothesis in *k*-best form.

4.2.2 Fully Hierarchical Chinese-to-English Translation. We translate Chinese-to-English with full hierarchical decoding: nonterminals are allowed to generate other hierarchical rules in recursion.

We apply the constraint *hmax*=10 for nonterminal category X, as described in Section 2.2.2, so that only glue rules are allowed at upper levels of the CYK grid; this is applied in both HCP and HiFST.

In HiFST any lattice in the CYK grid is pruned if it covers at least three source words and contains more than 10,000 states. The log-likelihood pruning threshold relative to the best path in the sublattices is 9.0.

Improved Optimization and Generalization Table 3 shows results for *tune-nw*, *test-nw*, and *mt08*. The first two rows show results for HCP when using MERT parameters optimized over *k*-best lists produced by HCP (row a) and by HiFST (row b); in the latter case, we are tuning HCP parameters over the hypothesis list generated by HiFST. When measured over *test-nw* this gives a 0.3 BLEU improvement. HCP benefits from tuning

Table 3
Contrastive Chinese-to-English translation results (lower-cased IBM BLEU) after first-pass decoding and subsequent rescoring steps. The MERT *k*-best column indicates which decoder generated the *k*-best lists used in MERT optimization. The *mt08* set contains 691 sentences of newswire and 666 sentences of Web text.

	decoder	MERT *k*-best	*tune-nw*	*test-nw*	*mt08*
a	HCP	HCP	32.8	33.1	–
b	HCP		32.9	33.4	28.2
	+5g	HiFST	33.4	33.8	28.7
	+5g+MBR		33.6	34.0	28.9
c	HiFST		33.1	33.4	28.1
	+5g	HiFST	33.8	34.3	29.0
	+5g+LMBR		34.5	34.9	30.2

over the HiFST hypotheses and we conclude that using the *k*-best list obtained by the HiFST decoder yields better parameters in optimization.

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

Lattice/k-best Quality Large LMs and MBR both benefit from the richer space of translation hypotheses contained in the lattices. Relative to the first-pass baseline in MT08, rescoring HiFST lattices yields a gain of 2.1 BLEU, compared to a gain of 0.7 BLEU with HCP *k*-best lists.

4.2.3 Reliability of n-gram Posterior Distributions. MBR decoding under linear BLEU (Tromble et al. 2008) is driven mainly by the presence of high posterior *n*-grams in the lattice; the low posterior *n*-grams have poor discriminatory power. In the following experiment, we show that high posterior *n*-grams are more likely to be found in the references, and that using the full evidence space of the lattice is much better than even very large *k*-best lists for computing posterior probabilities. Let $\mathcal{N}_i = \{w_1, \ldots, w_{|\mathcal{N}_i|}\}$ denote the set of *n*-grams of order *i* in the first-pass translation 1-best, and let $\mathcal{R}_i = \{w_1, \ldots, w_{|\mathcal{R}_i|}\}$ denote the set of *n*-grams of order *i* in the union of the references. For confidence threshold β, let $\mathcal{N}_{i,\beta} = \{w \in \mathcal{N}_i \colon p(w|\mathcal{L}) \geq \beta\}$ denote the set of all *n*-grams in $\mathcal{N}_i$ with posterior probability greater than or equal to β, where $p(w|\mathcal{L})$ is the posterior probability of the *n*-gram w, that is, the sum of the posterior probabilities of all translations containing at least one occurrence of w. The precision at order *i* for threshold β is the proportion of *n*-grams in $\mathcal{N}_{i,\beta}$ found in the references:

$$p_{i,\beta} = \frac{|\mathcal{R}_i \cap \mathcal{N}_{i,\beta}|}{|\mathcal{N}_{i,\beta}|}. \tag{7}$$

The average per-sentence 4-gram precision at a range of posterior probability thresholds β is shown in Figure 12. The posterior probabilities are computed using either the full

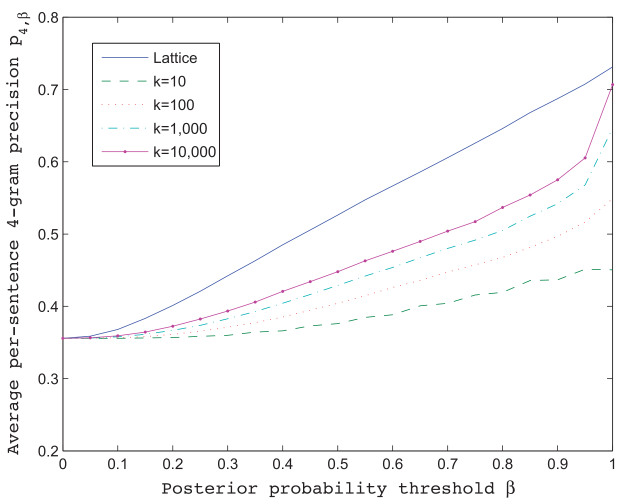


Figure 12
4-gram precisions for Arabic-to-English *mt02-05-tune* first-pass 1-best translations computed using the full evidence space of the lattice and *k*-best lists of various sizes.

lattice $\mathcal{L}$ or a *k*-best list of the specified size. The 4-gram precision of the 1-best translations is approximately 0.35. At higher values of β, the reference precision increases considerably. Expanding the *k*-best list size from 1,000 to 10,000 hypotheses only slightly improves the precision but much higher precisions are observed when the full evidence space of the lattice is used. The improved precision results from more accurate estimates of *n*-gram posterior probabilites and emphasizes once more the advantage of lattice-based decoding and rescoring techniques.

4.3 Grammar Configurations and Search Parameters

We report translation performance and decoding speed as we vary hierarchical grammar depth and the constraints on low-level rule concatenation (see Section 3). Unless otherwise noted, *hmin* = 1 and *hmax* = 10 throughout (except for the 'S' nonterminal category, where these constraints are not relevant).

4.3.1 Grammars for Arabic-to-English Translation. Table 4 reports Arabic-to-English translation results using the alternative grammar configurations described in Section 3.5. Results are shown in first-pass decoding (HiFST rows), and in rescoring with a larger 5-gram language model for the most promising configurations (+5g rows). Decoding time is reported for first-pass decoding only; rescoring time is negligible by comparison.

As shown in the upper part of Table 4, translation under a *shallow-2* grammar does not improve relative to a *shallow-1* grammar, although decoding is much slower. This indicates that the additional hypotheses generated when allowing a hierarchical depth of two are not useful in Arabic-to-English translation. By contrast the shallow grammars that allow long-distance movement for verbs only (shallow-1+$\mathrm{K}_1, \mathrm{K}_2 = 1, 3$, vo and shallow-2+$\mathrm{K}_1, \mathrm{K}_2 = 2, 3$, vo), perform slightly better than *shallow-1* grammar at a similar decoding time. Performance differences increase when the larger 5-gram is applied

Table 4
Contrastive Arabic-to-English translation results (lower-cased IBM BLEU) with various grammar configurations. Decoding time reported in seconds per word for *mt02-05-tune*.

	grammar	*time*	*mt02-05-tune*	*mt02-05-test*	*mt08*
HiFST	shallow-1	0.8	52.7	52.0	42.9
	+$\mathbf{K}_1, \mathbf{K}_2 = 1, 3$	1.3	52.6	51.9	42.8
	+$\mathbf{K}_1, \mathbf{K}_2 = 1, 3$, vo	0.9	52.7	52.1	42.9
	shallow-2	4.2	52.7	51.9	42.6
	+$\mathbf{K}_1, \mathbf{K}_2 = 2, 3$, vo	1.8	52.8	52.2	43.0
+5g	shallow-1	-	53.9	53.4	44.9
	+$\mathbf{K}_1, \mathbf{K}_2 = 1, 3$, vo	-	54.1	53.6	45.0
	shallow-2 +$\mathbf{K}_1, \mathbf{K}_2 = 2, 3$, vo	-	54.2	53.8	45.0

(Table 4, bottom). This is expected given that these grammars add valid translation candidates to the search space with similar costs; a language model is needed to select the good hypotheses among all those introduced.

4.3.2 Grammars for Chinese-to-English Translation. Table 5 shows contrastive results in Chinese-to-English translation for full hierarchical and shallow-n ($n = 1, 2, 3$) grammars.[3] Unlike Arabic-to-English translation, Chinese-to-English translation improves as the hierarchical depth of the grammar is increased (i.e., for larger n). Decoding time also increases significantly. The *shallow-1* grammar constraints which worked well for Arabic-to-English translation are clearly inadequate for this task; performance degrades by approximately 1.0 BLEU relative to the full hierarchical grammar.

However, we find that translation under the *shallow-3* grammar yields performance nearly as good as that of the full hiero grammars; translation times are shorter and yield degradations of only 0.1 to 0.3 BLEU. Translation can be made significantly faster by constraining the *shallow-3* search space with $hmin = 9, 5, 2$ for X^2,X^1, and X^0, respectively; translation speed is reduced from 10.8 sec/word to 3.8 sec/word at a degradation of 0.2 to 0.3 BLEU relative to full Hiero.

Shallow-3 grammars describe a restricted search-space but appear to have expressive power in Chinese-to-English translation which is very similar to that of a full Hiero grammar. Each cell (x, y) is represented by a bigger set of nonterminals; this allows for more effective pruning strategies during lattice construction. We note also that *hmax* values greater than 10 yield little improvement. As shown in the five bottom rows of Table 5, differences between grammar configurations tend to carry through after 5-gram rescoring. In summary, a *shallow-3* grammar and filtering with $hmin = 9, 5, 2$ lead to a 0.4 degradation in BLEU relative to full Hiero. As a final contrast, the mixed-case NIST BLEU-4 for the HiFST system on *mt08* is 28.6. This result is obtained under the same evaluation conditions as the official NIST MT08 Constrained Training Track.[4]

3 We note that the scores in full hiero row do not match those of row c in Table 3 which were obtained with a slightly simplified version of HiFST and optimized according to the 2008 NIST implementation of IBM BLEU; here we use the 2009 implementation by NIST.

4 Full MT08 results are available at `www.nist.gov/speech/tests/mt/2008/doc/mt08_official_results_v0.html`

Table 5
Contrastive Chinese-to-English translation results (lower-cased IBM BLEU) with various grammar configurations and search parameters. Decoding time is reported in sec/word for *tune-nw*.

	grammar	*time*	*tune-nw*	*test-nw*	*mt08 (nw)*
HiFST	shallow-1	0.7	33.6	33.4	32.6
	shallow-2	5.9	33.8	34.2	32.7
	+*hmin*=5	5.6	33.8	34.1	32.9
	+*hmin*=7	4.0	33.8	34.3	33.0
	shallow-3	8.8	34.0	34.3	33.0
	+*hmin*=7	7.7	34.0	34.4	33.1
	+*hmin*=9	5.9	33.9	34.3	33.1
	+*hmin*=9,5,2	3.8	34.0	34.3	33.0
	+*hmin*=9,5,2+*hmax*=11	6.1	33.8	34.4	33.0
	+*hmin*=9,5,2+*hmax*=13	9.8	34.0	34.4	33.1
	full hiero	10.8	34.0	34.4	33.3
+5g	shallow-1	-	34.1	34.5	33.4
	shallow-2	-	34.3	35.1	34.0
	shallow-3	-	34.6	35.2	34.4
	+*hmin*=9,5,2	-	34.5	34.8	34.2
	full hiero	-	34.5	35.2	34.6

4.4 Marginalization Over Translation Derivations

As has been discussed earlier, the translation model in hierarchical phrase-based machine translation allows for multiple derivations of a target language sentence. Each derivation corresponds to a particular combination of hierarchical rules and it has been argued that the correct approach to translation is to accumulate translation probability by summing over the scores of all derivations (Blunsom, Cohn, and Osborne 2008). Computing this sum for each of the many translation candidates explored during decoding is computationally difficult, however. For this reason the translation probability is commonly computed using the Viterbi max-derivation approximation. This is the approach taken in the previous sections in which translations scores were accumulated under the tropical semiring.

The use of WFSTs allows the sum over alternative derivations of a target string to be computed efficiently. HiFST generates a translation lattice realized as a weighted transducer with output labels encoding words and input labels encoding the sequence of rules corresponding to a particular derivation, and the cost of each path in the lattice is the negative log probability of the derivation that generated the hypothesis.

Determinization applies the $\oplus$ operator to all paths with the same word sequence (Mohri 1997). When applied in the log semiring, this operator computes the sum of two paths with the same word sequence as $x \oplus y = -\log(e^{-x} + e^{-y})$ so that the probabilities of alternative derivations can be summed.

Currently this operation is only performed in the top cell of the hierarchical decoder so it is still an approximation to the true translation probability. Computing the true translation probability would require the same operation to be repeated in every cell during decoding, which is very time consuming. Note that the translation lattice was generated with a language model and so the language model costs must be removed

Table 6
Arabic-to-English results (lower-cased IBM BLEU) when determinizing the lattice at the upper-most CYK cell with alternative semirings.

semiring		*mt02-05-tune*	*mt02-05-test*	*mt08*
tropical	HiFST	52.8	52.2	43.0
	+5g	54.2	53.8	44.9
	+5g+LMBR	55.0	54.6	45.5
log	HiFST	53.1	52.6	43.2
	+5g	54.6	54.2	45.2
	+5g+LMBR	55.0	54.6	45.5

before determinization to ensure that only the derivation probabilities are included in the sum. After determinization, the language model is reapplied and the 1-best translation hypothesis can be extracted from the logarc determinized lattices.

Table 6 compares translation results obtained using the tropical semiring (Viterbi likelihoods) and the log semiring (marginal likelihoods). First-pass translation shows small gains in all sets: +0.3 and +0.4 BLEU for *mt02-05-tune* and *mt02-05-test*, and +0.2 for *mt08*. These gains show that the sum over alternative derivations can be easily obtained in HiFST simply by changing semiring and that these alternative derivations are beneficial to translation. The gains carry through to the large language model 5-gram rescoring stage but after LMBR the final BLEU scores are unchanged. The hypotheses selected by LMBR are in almost all cases exactly the same regardless of the choice of semiring. This may be due to the fact that our current marginalization procedure is only an approximation to the true marginal likelihoods, since the log semiring determinization operation is applied only in the uppermost cell of the CYK grid and MERT training is performed using regular Viterbi likelihoods.

We note that a close study of the interaction between LMBR and marginalization over derivations is beyond the scope of this paper. Our purpose here is to show how easily these operations can be done using WFSTs.

4.5 Combining Lattices Obtained from Alternative Morphological Decompositions

It has been shown that MBR decoding is a very effective way of combining translation hypotheses obtained from alternative morphological decompositions of the same source data. In particular, de Gispert et al. (2009) show gains for Arabic-to-English and Finnish-to-English when taking k-best lists obtained from two morphological decompositions of the source language. Here we extend this approach to the case of translation lattices and experiment with more than two alternative decompositions. We will show that working with translation lattices gives significant improvements relative to k-best lists.

In lattice-based MBR system combination, first-pass decoding results in a set of I distinct translation lattices $\mathcal{L}^{(i)},\ i = 1 \ldots I$ for each foreign sentence, with each lattice produced by translating one of the alternative morphological decompositions. The evidence space for MBR decoding is formed as the union of these lattices $\mathcal{L} = \bigcup_{i=1}^{I} \mathcal{L}^{(i)}$. The posterior probability of n-gram w in the union of lattices is computed as a simple

Table 7
Arabic-to-English results (lower-cased IBM BLEU) when using alternative Arabic decompositions, and their combination with *k*-best-based and lattice-based MBR.

configuration		*mt02-05-tune*	*mt02-05-test*	*mt08*
a	HiFST+5g	54.2	53.8	44.9
b	HiFST+5g	53.8	53.6	45.0
c	HiFST+5g	54.1	53.8	44.7
a+b	+MBR	55.1	54.7	46.1
	+LMBR	55.7	55.4	46.7
a+c	+MBR	55.4	54.9	46.5
	+LMBR	56.0	55.9	46.9
a+b+c	+MBR	55.3	54.9	46.5
	+LMBR	56.0	55.7	47.3

linear interpolation of the posterior probabilities according to the evidence space of each individual lattice so that

$$p(w|\mathcal{L}) = \sum_{i=1}^{I} \lambda_i \, p_i(w|\mathcal{L}^{(i)}), \tag{8}$$

where the interpolation parameters $0 \leq \lambda_i \leq 1$ such that $\sum_{i=1}^{I} \lambda_i = 1$ specify the weight associated with each system in the combination and are optimized with respect to the tuning set. The system-specific posteriors required for the interpolation are computed as

$$p_i(w|\mathcal{L}^{(i)}) = \sum_{E \in \mathcal{L}_w^{(i)}} P_i(E|F), \tag{9}$$

where $P_i(E|F)$ is the posterior probability of translation E given source sentence F and the sum is taken over the subset $\mathcal{L}_w^{(i)} = \{E \in \mathcal{L}^{(i)} \colon \#_w(E) > 0\}$ of the lattice $\mathcal{L}^{(i)}$ containing paths with at least one occurrence of the n-gram w. These posterior probabilities are used in MBR decoding under the linear approximation to the BLEU score described in Tromble et al. (2008). We find that for system combination, decoding often produces output that is slightly shorter than required. A fixed per-word factor optimized on the tuning set is applied when computing the gain and this results in output with improved BLEU score and reduced brevity penalty.

Table 7 shows translation results in Arabic-to-English using three alternative morphological decompositions of the Arabic text (upper rows a, b, and c). For each decomposition an independent set of hierarchical rules is obtained from the respective parallel corpus alignments. The decompositions were generated by the MADA toolkit (Habash and Rambow 2005) with two alternative tokenization schemes, and by the Sakhr Arabic Morphological Tagger, developed by Sakhr Software in Egypt.

The following rows show the results when combining with MBR the translation hypotheses obtained from two or three decompositions. The table also shows a contrast

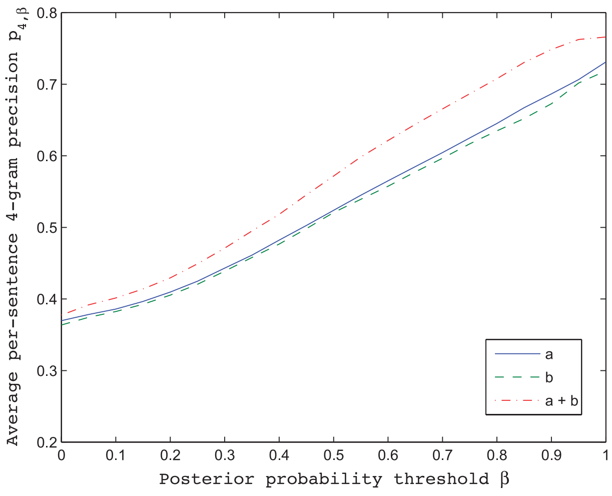


Figure 13
Average per-sentence 4-gram reference precisions for Arabic-to-English *mt02-05-tune* single-system MBR 1-best translations and the 1-best obtained through MBR system combination.

between decoding the joint k-best lists (rows named MBR, with $k = 1,000$) and decoding the unioned translation lattices (rows named LMBR). In line with the findings of de Gispert et al. (2009), we find significant gains from combining k-best lists with respect to using any one segmentation alone. Interestingly, here we find further gains when applying lattice-based MBR instead of a k-best approach, obtaining consistent gains of 0.6–0.8 BLEU across all sets.

The results reported in Table 7 are very competitive. The mixed-case NIST BLEU-4 score for a+b+c LMBR system in MT08 is 44.9. This result is obtained under the same evaluation conditions as the official NIST MT08 Constrained Training Track.[5] For MT09, the mixed-case BLEU-4 is 48.3, which ranks first in the Arabic-to-English NIST 2009 Constrained Data Track.[6]

4.5.1 System Combination and Reference Precision. We have demonstrated that MBR decoding of multiple lattices generated from alternative morphological segmentations leads to significant improvements in BLEU score. We now show that one reason for the improved performance is that lattice combination leads to better n-gram posterior probability estimates. To combine two equally weighted lattices $\mathcal{L}^{(1)}$ and $\mathcal{L}^{(2)}$, the interpolation weights are $\lambda_1 = \lambda_2 = \frac{1}{2}$; Equation (8) simplifies as $p(w|\mathcal{L}) = \frac{1}{2}(p_1(w|\mathcal{L}^{(1)}) + p_2(w|\mathcal{L}^{(2)}))$. Figure 13 plots average per-sentence reference precisions for the 4-grams in the MBR 1-best of systems a and b and their combination (labeled a+b) at a range of posterior probability thresholds $0 \leq \beta \leq 1$. Systems a and b have similar precisions at all values of β, confirming that the optimal interpolation weights for this combination should be equal. The precision obtained using n-gram posterior probabilities computed

5 Full MT08 results are available at
`www.nist.gov/speech/tests/mt/2008/doc/mt08_official_results_v0.html`
6 Full MT09 results are available at `www.itl.nist.gov/iad/mig/tests/mt/2009/ResultsRelease`

from the combined lattices is higher than that of the individual systems. A higher proportion of the *n*-grams assigned high posterior probability under the interpolated distribution are found in the references and this is one of the reasons for the large gains in BLEU in lattice-based MBR system combination.

4.6 European Language Translation

The HiFST described here has also been found to achieve competitive performance for other language pairs, such as Spanish-to-English and Finnish-to-English.

For Spanish-to-English we carried out experiments on the shared task of the ACL 2008 Workshop on Statistical Machine Translation (Callison-Burch et al. 2008) based on the Europarl corpus. For the official *test2008* evaluation set we obtain a BLEU score of 34.2 using a shallow-1 grammar. Similarly to the Arabic case, deeper grammars are not found to improve scores for this task.

In Finnish-to-English, we conducted translation experiments based on the Europarl corpus using 3,000 sentences from the Q4/2000 period for testing with a single reference. In this case, the shallow-1 grammar obtained a BLEU score of 28.0, whereas the full hierarchical grammar only achieved 27.6. This is further evidence that full-hierarchical grammars are not appropriate in all instances. In this case we suspect that the use of Finnish words without morphological decomposition leads to data sparsity problems and complicates the task of learning complex translation rules. The lack of a large English language model suitable for this domain may also make it harder to select the right hypothesis when the translation grammar produces many more English alternatives.

5. Conclusions

We have described two linked investigations into hierarchical phrase-based translation. We investigate the use of weighted finite state transducers rather than *k*-best lists to represent the space of translation hypotheses. We describe a lattice-based Hiero decoder, with which we find reductions in search errors, better parameter optimization, and improved translation performance. Relative to these reductions in search errors, direct generation of target language translation lattices also leads to further translation improvements through subsequent rescoring steps, such as MBR decoding and the application of large *n*-gram language models. These steps can be carried out easily via standard WFST operations.

As part of the machinery needed for our experiments we develop WFST procedures for alignment and feature extraction so that statistics needed for system optimization can be easily obtained and represented as transducers. In particular, we make use of a lattice-based representation of sequences of rule applications, which proves useful for minimum error rate training. In all instances we find that using lattices as compact representations of translation hypotheses offers clear modeling advantages.

We also investigate refinements in translation search space through shallow-*n* grammars, structured long-distance movement, and constrained word deletion. We find that these techniques can be used to fit the complexity of Hiero translation systems to individual language pairs. In translation from Arabic into English, shallow grammars make it possible to explore the entire search space and to do so more quickly but with the same translation quality as the full Hiero grammar. Even in complex translation tasks, such as Chinese to English, we find significant speed improvements with minimal loss in performance using these methods. We take the view that it is better to perform exact search of a constrained space than to risk search errors in translation.

We note finally that Chiang introduced Hiero as a model "based on a synchronous CFG, elsewhere known as a syntax-directed transduction grammar (Lewis and Stearns 1968)." We have taken this formulation as a starting point for the development of novel realizations of Hiero. Our motivation has mainly been practical in that we seek improved translation quality and efficiency through better models and algorithms.

Our approach suggests close links between Hiero and Recursive Transition Networks (Woods 1970; Mohri 1997). Although this connection is beyond the scope of this paper, we do note that Hiero translation requires keeping track of two grammars, one based on the Hiero translation rules and the other based on *n*-gram language model probabilities. These two grammars have very different dependencies which suggests that a full implementation of Hiero translation such as we have addressed does not have a simple expression as an RTN.

Acknowledgments

This work was supported in part by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011- 06-C-0022, and in part by the Spanish government and the ERDF under projects TEC2006-13694-C03-03 and TEC2009-14094-C04-04.

References

Allauzen, Cyril, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of ACL*, pages 557–564, Sapporo.

Allauzen, Cyril, 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, Prague.

Bender, Oliver, 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, Kyoto.

Blunsom, Phil, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-HLT*, pages 200–208, Columbus, OH.

Brants, Thorsten, 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, Prague.

Callison-Burch, Chris, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, pages 70–106, Columbus, OH.

Chappelier, Jean-Cédric and Martin Rajman. 1998. A generalized CYK algorithm for parsing stochastic CFG. In *Proceedings of TAPD*, pages 133–137, Paris.

Chiang, David. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270, Ann Arbor, MI.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

de Gispert, Adrià, Sami Virpioja, Mikko Kurimo, and William Byrne. 2009. Minimum Bayes-Risk combination of translation hypotheses from alternative morphological decompositions. In *Proceedings of HLT/NAACL, Companion Volume: Short Papers*, pages 73–76, Boulder, CO.

Deng, Yonggang and William Byrne. 2008. HMM word and phrase alignment for statistical machine translation. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):494–507.

Graehl, Jonathan, Kevin Knight, and Jonathan May. 2008. Training tree transducers. *Computational Linguistics*, 34(3):391–427.

Habash, Nizar and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the ACL*, pages 573–580, Ann Arbor, MI.

Iglesias, Gonzalo, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009a. Rule filtering by pattern for efficient hierarchical translation. In *Proceedings of the EACL*, pages 380–388, Athens.

Iglesias, Gonzalo, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009b. Hierarchical phrase-based translation with weighted finite state transducers. In *Proceedings of HLT/NAACL*, pages 433–441, Boulder, CO.

Iglesias, Gonzalo, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009c. The HiFST system for the Europarl Spanish-to-English task. In *Proceedings of SEPLN*, pages 207–214, Donosti.

Knight, Kevin and Yaser Al-Onaizan. 1998. Translation with finite-state devices. In *Proceedings of the Third Conference of the AMTA on Machine Translation and the Information Soup*, pages 421–437, Langhorne, PA.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133, Edmonton.

Kumar, Shankar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of HLT-NAACL*, pages 169–176, Boston, MA.

Kumar, Shankar, Yonggang Deng, and William Byrne. 2006. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*, 12(1):35–75.

Lewis, P. M., II, and R. E. Stearns. 1968. Syntax-directed transduction. *Journal of the ACM*, 15(3):465–488.

Mohri, Mehryar. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:269–311.

Mohri, Mehryar, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231:17–32.

Mohri, Mehryar, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16:69–88.

Och, Franz J. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167, Sapporo.

Och, Franz Josef and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the ACL*, pages 295–302, Philadelphia, PA.

Papineni, Kishore, 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, Toulouse.

Rosti, Antti-Veikko, 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, Rochester, NY.

Setiawan, Hendra, Min Yen Kan, Haizhou Li, and Philip Resnik. 2009. Topological ordering of function words in hierarchical phrase-based translation. In *Proceedings of the ACL-IJCNLP*, pages 324–332, Singapore.

Sim, Khe Chai, William Byrne, Mark Gales, Hichem Sahbi, and Phil Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *Proceedings of ICASSP*, pages 105–108, Honolulu, HI.

Tromble, Roy, 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, Honolulu, HI.

Varile, Giovanni B. and Peter Lau. 1988. Eurotra practical experience with a multilingual machine translation system under development. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 160–167, Austin, TX.

Woods, W. A. 1970. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606.

Wu, Dekai. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.