Extraction Programs: A Unified Approach to Translation Rule Extraction

Mark Hopkins and Greg Langmead and Tai Vo
SDL Language Technologies Division
6060 Center Drive, Suite 150
Los Angeles, CA 90045
{mhopkins,glangmead,tvo}@sdl.com

Abstract

We provide a general algorithmic schema for translation rule extraction and show that several popular extraction methods (including phrase pair extraction, hierarchical phrase pair extraction, and GHKM extraction) can be viewed as specific instances of this schema. This work is primarily intended as a survey of the dominant extraction paradigms, in which we make explicit the close relationship between these approaches, and establish a language for future hybridizations. This facilitates a generic and extensible implementation of alignment-based extraction methods.

1 Introduction

The tradition of extracting translation rules from aligned sentence pairs dates back more than a decade. A prominent early example is phrase-based extraction (Och et al., 1999).

Around the middle of the last decade, two extraction paradigms were proposed for syntax-based machine translation: the Hiero paradigm of (Chiang, 2005) and the GHKM paradigm of (Galley et al., 2004). From these papers followed two largely independent lines of research, respectively dubbed *formally syntax-based machine translation* (Chiang, 2007; Zollmann and Venugopal, 2006; Venugopal et al., 2007; Lopez, 2007; Marton and Resnik, 2008; Li et al., 2009; de Gispert et al., 2010) and *linguistically syntax-based machine translation* (Galley et al., 2006; Marcu et al., 2006; Liu et al., 2006; Huang et al., 2006; Liu et al., 2007; Mi and Huang, 2008; Zhang et al., 2008; Liu et al., 2009).

In this paper, we unify these strands of research by showing how to express Hiero extraction, GHKM extraction, and phrase-based extraction as instances of a single master extraction method. Specifically, we express each technique as a simple "program" given to a generic "evaluator". Table 1 summarizes how to express several popular extraction methods as "extraction programs."

Besides providing a unifying survey of popular alignment-based extraction methods, this work has the practical benefit of facilitating the implementation of these methods. By specifying the appropriate input program, the generic evaluator (coded, say, as a Python module) can be used to execute any of the extraction techniques in Table 1. New extraction techniques and hybridizations of existing techniques can be supported with minimal additional programming.

2 Building Blocks

The family of extraction algorithms under consideration share a common setup: they extract translation rules from a sentence pair and an alignment. In this section, we define these concepts.

2.1 Patterns and Sentences

Assume we have a global vocabulary of atomic *symbols*, containing the reserved *substitution symbol* ∇. Define a *pattern* as a sequence of symbols. Define the *rank* of a pattern as the count of its ∇ symbols.

Let $\nabla^k \triangleq \langle\overbrace{\nabla, \nabla, ..., \nabla}^{k}\rangle$.

We will typically use space-delimited quotations to represent example patterns, e.g. "ne ∇ pas" rather than $\langle \text{ne}, \nabla, \text{pas}\rangle$. We will use the dot operator to represent the concatenation of patterns, e.g. "il ne" $\cdot$ "va pas" = "il ne va pas".

	Extraction Program		
Method	Primary Protocol	Secondary Protocol	Labeling Protocol
PBMT (Och et al., 1999)	$\textsc{RankPP}_0$	$\textsc{TrivSP}_{\mathcal{A}}$	$\textsc{TrivLP}$
Hiero (Chiang, 2005)	$\textsc{RankPP}_\infty$	$\textsc{TrivSP}_{\mathcal{A}}$	$\textsc{TrivLP}$
GHKM (Galley et al., 2004)	$\textsc{MapPP}_t$	$\textsc{TrivSP}_{\mathcal{A}}$	$\textsc{PMapLP}_t$
SAMT (Zollmann and Venugopal, 2006)	$\textsc{RankPP}_\infty$	$\textsc{TrivSP}_{\mathcal{A}}$	$\textsc{PMapLP}_{\tilde{t}}$
Forest GHKM (Mi and Huang, 2008)	$\textsc{MapPP}_T$	$\textsc{TrivSP}_{\mathcal{A}}$	$\textsc{PMapLP}_T$
Tree-to-Tree GHKM (Liu et al., 2009)	$\textsc{MapPP}_t$	$\textsc{MapSP}_{\tau,\mathcal{A}}$	$\textsc{IMapLP}_{\{t\},\{\tau\}}$
Forest-to-Forest GHKM (Liu et al., 2009)	$\textsc{MapPP}_T$	$\textsc{MapSP}_{\mathcal{T},\mathcal{A}}$	$\textsc{IMapLP}_{T,\mathcal{T}}$
Fuzzy Dual Syntax (Chiang, 2010)	$\textsc{MapPP}_{\tilde{t}}$	$\textsc{MapSP}_{\tilde{\tau},\mathcal{A}}$	$\textsc{IMapLP}_{\{\tilde{t}\},\{\tilde{\tau}\}}$

Table 1: Various rule extraction methods, expressed as extraction programs. Boldfaced methods are proven in this paper; the rest are left as conjecture. Parameters: t, τ are spanmaps (see Section 3); $\tilde{t}, \tilde{\tau}$ are fuzzy spanmaps (see Section 7); $T, \mathcal{T}$ are sets of spanmaps (typically encoded as forests); $\mathcal{A}$ is an alignment (see Section 2).

We refer to a contiguous portion of a pattern with a *span*, defined as either the *null span* ϕ , or a pair $[b, c]$ of positive integers such that $b \leq c$. We will treat span $[b, c]$ as the implicit encoding of the set $\{b, b+1, ..., c\}$, and employ set-theoretic operations on spans, e.g. $[3, 8] \cap [6, 11] = [6, 8]$. Note that the null span encodes the empty set.

If a set I of positive integers is non-empty, then it has a unique *minimal enclosing span*, defined by the operator $\mathsf{span}(I) = [\mathsf{min}(I), \mathsf{max}(I)]$. For instance, $\mathsf{span}(\{1, 3, 4\}) = [1, 4]$. Define $\mathsf{span}(\{\}) = \phi$.

Finally, define a *sentence* as a pattern of rank 0.

2.2 Alignments

An *alignment* is a triple $\langle m, n, \mathfrak{A} \rangle$, where m and n are positive integers, and $\mathfrak{A}$ is a set of ordered integer pairs (i, j) such that $1 \leq i \leq m$ and $1 \leq j \leq n$.

In Figure 1(a), we show a graphical depiction of alignment $\langle 4, 6, \{(1, 1), (2, 3), (4, 3), (3, 5)\} \rangle$. Observe that alignments have a *primary* side (top) and a *secondary* side (bottom)[1]. For alignment $\mathcal{A} = \langle m, n, \mathfrak{A} \rangle$, define $|\mathcal{A}|_p = m$ and $|\mathcal{A}|_s = n$. A *primary index* (resp., *secondary index*) of $\mathcal{A}$ is any positive integer less than or equal to $|\mathcal{A}|_p$ (resp., $|\mathcal{A}|_s$). A *primary span* (resp., *secondary span*) of $\mathcal{A}$ is any span $[b, c]$ such that $1 \leq b \leq c \leq |\mathcal{A}|_p$ (resp., $|\mathcal{A}|_s$).

Define $a \overset{\mathcal{A}}{\sim} \alpha$ to mean that $(a, \alpha) \in \mathfrak{A}$ (in words, we say that $\mathcal{A}$ *aligns* primary index a to secondary index α), and define $a \overset{\mathcal{A}}{\not\sim} \alpha$ to mean that $(a, \alpha) \notin \mathfrak{A}$.

[1]The terms *primary* and *secondary* allow us to be agnostic about how the extracted rules are used in a translation system, i.e. the primary side can refer to the source or target language.

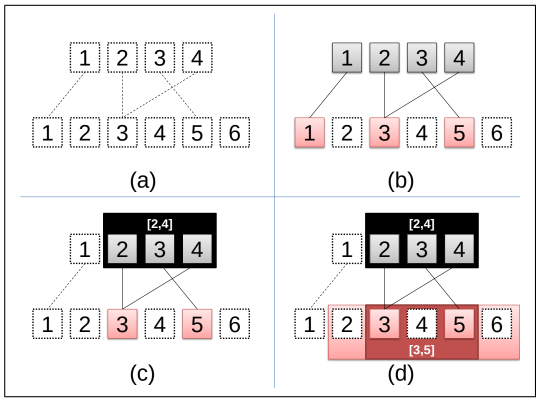


Figure 1: A demonstration of alignment terminology. (a) An alignment is a relation between positive integer sets. (b) The *primary domain* of the example alignment is {1,2,3,4} and the *secondary domain* is {1,3,5}. (c) The *image* of primary span [2,4] is {3,5}. (d) The *minimal projection* of primary span [2,4] is [3,5]. Secondary spans [2,5], [3,6], and [2,6] are also *projections* of primary span [2,4].

Define an *aligned sentence pair* as a triple $\langle s, \sigma, \mathcal{A} \rangle$ where $\mathcal{A}$ is an alignment and s, σ are sentences of length $|\mathcal{A}|_p$ and $|\mathcal{A}|_s$, respectively.

Primary and Secondary Domain: The *primary domain* of alignment $\mathcal{A}$ is the set of primary indices that are aligned to some secondary index, i.e. $\mathsf{pdom}(\mathcal{A}) = \{a | \exists \alpha \text{ s.t. } a \overset{\mathcal{A}}{\sim} \alpha\}$. Analogously, define $\mathsf{sdom}(\mathcal{A}) = \{\alpha | \exists a \text{ s.t. } a \overset{\mathcal{A}}{\sim} \alpha\}$. For the example alignment of Figure 1(b), $\mathsf{pdom}(\mathcal{A}) =$

$\{1,2,3,4\}$ and $\mathsf{sdom}(\mathcal{A}) = \{1,3,5\}$.

Image: The *image* of a set I of primary indices (denoted $\mathsf{pimage}_{\mathcal{A}}(I)$) is the set of secondary indices to which the primary indices of I align. In Figure 1(c), for instance, the image of primary span $[2,4]$ is the set $\{3,5\}$. Formally, for a set I of primary indices of alignment $\mathcal{A}$, define:

$$\mathsf{pimage}_{\mathcal{A}}(I) = \{\alpha | \exists a \in I \text{ s.t. } (a, \alpha) \in \mathfrak{A}\}$$

Projection: The *minimal projection* of a set I of primary indices (denoted $\mathsf{pmproj}_{\mathcal{A}}(I)$) is the minimal enclosing span of the image of I. In other words, $\mathsf{pmproj}_{\mathcal{A}}(I) = \mathsf{span}(\mathsf{pimage}_{\mathcal{A}}(I))$. In Figure 1(d), for instance, the minimal projection of primary span $[2,4]$ is the secondary span $[3,5]$.

Consider Figure 1(d). We will also allow a more relaxed type of projection, in which we allow the broadening of the minimal projection to include unaligned secondary indices. In the example, secondary spans $[2,5]$, $[3,6]$, and $[2,6]$ (in addition to the minimal projection $[3,5]$) are all considered *projections* of primary span $[2,4]$. Formally, define $\mathsf{pproj}_{\mathcal{A}}([b,c])$ as the set of superspans $[\beta,\gamma]$ of $\mathsf{pmproj}_{\mathcal{A}}([b,c])$ such that $[\beta,\gamma] \cap \mathsf{sdom}(\mathcal{A}) \subseteq \mathsf{pmproj}_{\mathcal{A}}([b,c])$.

2.3 Rules

We define an *unlabeled rule* as a tuple $\langle k, s^*, \sigma^*, \pi \rangle$ where k is a nonnegative integer, s^* and σ^* are patterns of rank k, and π is a permutation of the sequence $\langle 1, 2, ..., k \rangle$. Such rules can be rewritten using a more standard Synchronous Context-Free Grammar (SCFG) format, e.g. $\langle 3$, "le ∇ ∇ de ∇", "∇ 's ∇ ∇", $\langle 3,2,1\rangle\rangle$ can be written: $\nabla \rightarrow \langle$le ∇_1 ∇_2 de ∇_3, ∇_3 's ∇_2 $\nabla_1\rangle$.

A *labeled rule* is a pair $\langle r, l\rangle$, where r is an unlabeled rule, and l is a "label". The unlabeled rule defines the essential structure of a rule. The label gives us auxiliary information we can use as decoding constraints or rule features. This deliberate modularization lets us unify sequence-based and tree-based extraction methods.

Labels can take many forms. Two examples (depicted in Figure 2) are:

1. An *SCFG label* is a $(k+1)$-length sequence of symbols.

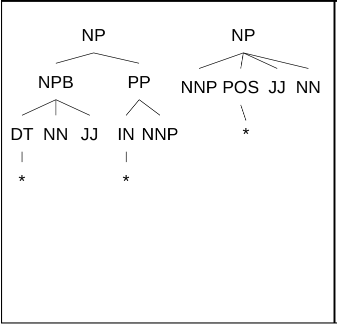
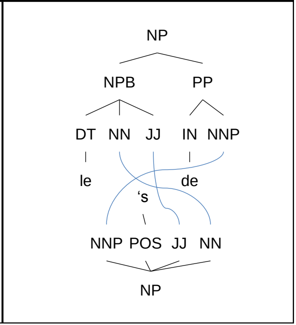
label	labeled rule
< NP, NN, JJ, NNP >	NP → < le NN_1 JJ_2 de NNP_3, NNP_3 's JJ_2 NN_1 >
	

Figure 2: An example SCFG label (top) and STSG label (bottom) for unlabeled rule $\nabla \rightarrow \langle$le ∇_1 ∇_2 de ∇_3, ∇_3 's ∇_2 $\nabla_1\rangle$.

2. An *STSG label* (from Synchronous Tree Substitution Grammar (Eisner, 2003)) is a pair of trees.

STSG labels subsume SCFG labels. Thus STSG extraction techniques can be used as SCFG extraction techniques by ignoring the extra hierarchical structure of the STSG label. Due to space constraints, we will restrict our focus to SCFG labels. When considering techniques originally formulated to extract STSG rules (GHKM, for instance), we will consider their SCFG equivalents.

3 A General Rule Extraction Schema

In this section, we develop a general algorithmic schema for extracting rules from aligned sentence pairs. We will do so by generalizing the GHKM algorithm (Galley et al., 2004). The process goes as follows:

- Repeatedly:
  - Choose a "construction request," which consists of a "primary subrequest" (see Figure 3a) and a "secondary subrequest" (see Figure 3b).
  - Construct the unlabeled rule corresponding to this request (see Figure 3, bottom).
  - Label the rule (see Figure 2).

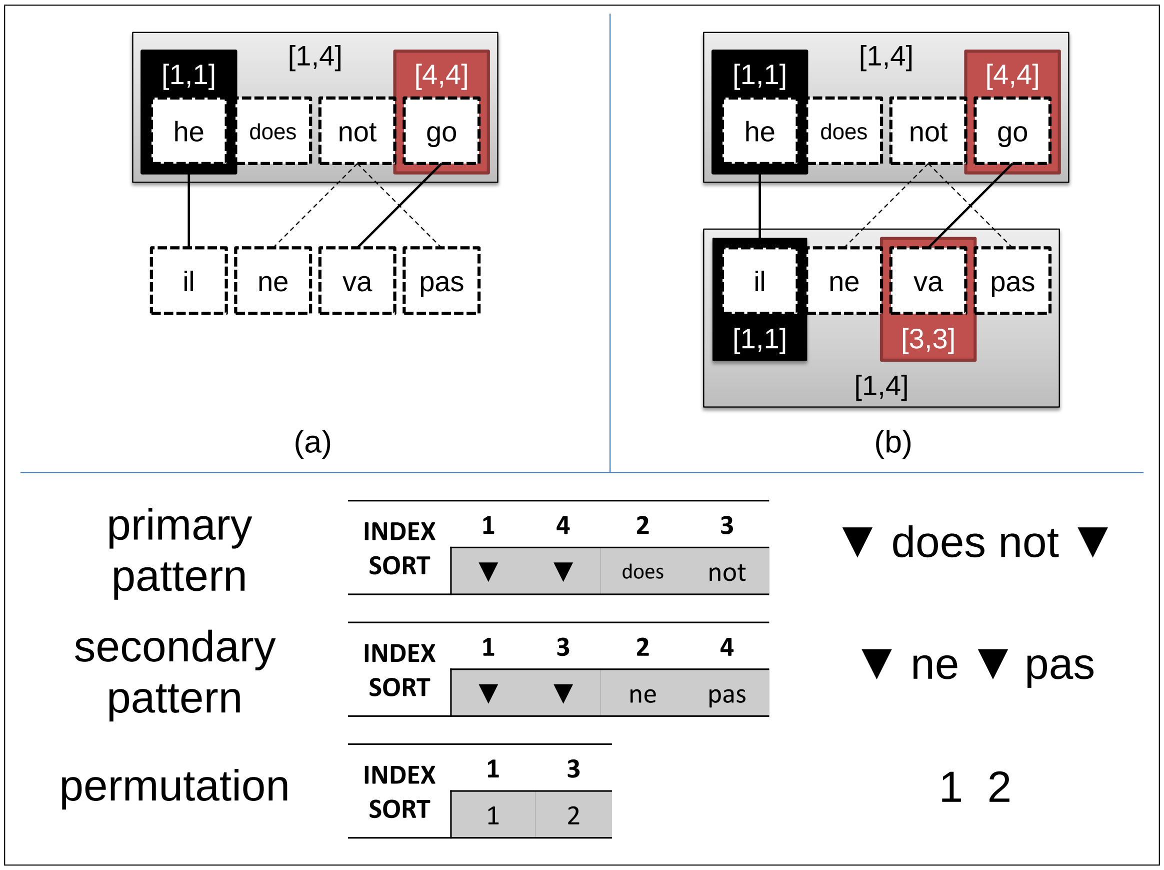


Figure 3: Extraction of the unlabeled rule $\nabla \rightarrow \langle \nabla_1$ does not ∇_2, ∇_1 ne ∇_2 pas$\rangle$. (a) Choose primary subrequest $[1,4] \rightsquigarrow [1,1][4,4]$. (b) Choose secondary subrequest $[1,4] \rightsquigarrow [1,1][3,3]$. (bottom) Construct the rule $\nabla \rightarrow \langle \nabla_1$ does not ∇_2, ∇_1 ne ∇_2 pas$\rangle$.

3.1 Choose a Construction Request

The first step in the extraction process is to choose a "construction request," which directs the algorithm about which unlabeled rule(s) we wish to construct. A "construction request" consists of two "subrequests."

Subrequests: A *subrequest* is a nonempty sequence of non-null spans $\langle [b_0, c_0], [b_1, c_1], ..., [b_k, c_k] \rangle$ such that, for all $1 \leq i < j \leq k$, $[b_i, c_i]$ and $[b_j, c_j]$ are disjoint proper[2] subsets of $[b_0, c_0]$. If it also true that $c_i < b_j$, for all $1 \leq i < j \leq k$, then the subrequest is called *monotonic*. We refer to k as the *rank* of the subrequest.

We typically write subrequest $\langle [b_0, c_0], [b_1, c_1], ..., [b_k, c_k] \rangle$ using the notation:

$$[b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_k, c_k]$$

or as $[b_0, c_0] \rightsquigarrow \epsilon$ if $k = 0$.

For subrequest $x = [b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_k, c_k]$, define:

$$\mathsf{covered}(x) = \cup_{i=1}^{k} [b_i, c_i]$$
$$\mathsf{uncovered}(x) = [b_0, c_0] \backslash \mathsf{covered}(x)$$

Primary Subrequests: Given an alignment $\mathcal{A}$, define the set $\mathsf{frontier}(\mathcal{A})$ as the set of primary spans $[b, c]$ of alignment $\mathcal{A}$ such that $\mathsf{pmproj}_{\mathcal{A}}([b, c])$ is nonempty and disjoint from $\mathsf{pimage}_{\mathcal{A}}([1, b-1]) \cup \mathsf{pimage}_{\mathcal{A}}([c+1, |\mathcal{A}|_p])$.[3]

[2]If unary rules are desired, i.e. rules of the form $\nabla \rightarrow \nabla$, then this condition can be relaxed.

[3]Our definition of the frontier property is an equivalent reexpression of that given in (Galley et al., 2004). We reexpress it in these terms in order to highlight the fact that the frontier

```
**Algorithm** CONSTRUCTRULE$_{s,\sigma,\mathcal{A}}(x, \xi)$:
  **if** construction request $\langle x, \xi \rangle$ matches alignment $\mathcal{A}$ **then**
    $\{u_1, ..., u_p\} = \mathsf{uncovered}([b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_k, c_k])$
    $\{v_1, ..., v_q\} = \mathsf{uncovered}([\beta_0, \gamma_0] \rightsquigarrow [\beta_1, \gamma_1]...[\beta_k, \gamma_k])$
    $s^* = \text{INDEXSORT}(\langle b_1, b_2, ..., b_k, u_1, u_2, ..., u_p \rangle, \langle \overbrace{\nabla, \nabla, ..., \nabla}^{k}, s_{u_1}, s_{u_2}, ..., s_{u_p} \rangle)$
    $\sigma^* = \text{INDEXSORT}(\langle \beta_1, \beta_2, ..., \beta_k, v_1, v_2, ..., v_q \rangle, \langle \overbrace{\nabla, \nabla, ..., \nabla}^{k}, \sigma_{v_1}, \sigma_{v_2}, ..., \sigma_{v_q} \rangle)$
    $\pi = \text{INDEXSORT}(\langle \beta_1, \beta_2, ..., \beta_k \rangle, \langle 1, 2, ..., k \rangle)$
    **return** $\{\langle k, s^*, \sigma^*, \pi \rangle\}$
  **else**
    **return** $\{\}$
  **end if**
```

Figure 4: Pseudocode for rule construction. Arguments: s = "$s_1\ s_2\ ...\ s_m$" and σ = "$\sigma_1\ \sigma_2\ ...\ \sigma_n$" are sentences, $\mathcal{A} = \langle m, n, \mathfrak{A} \rangle$ is an alignment, $x = [b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_k, c_k]$ and $\xi = [\beta_0, \gamma_0] \rightsquigarrow [\beta_1, \gamma_1]...[\beta_k, \gamma_k]$ are subrequests.

Define $\mathsf{preqs}(\mathcal{A})$ as the set of monotonic subrequests whose spans are all in $\mathsf{frontier}(\mathcal{A})$. We refer to members of $\mathsf{preqs}(\mathcal{A})$ as *primary subrequests* of alignment $\mathcal{A}$. Figure 3a shows a primary subrequest of an example alignment.

Secondary Subrequests: Given a primary subrequest $x = [b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_k, c_k]$ of alignment $\mathcal{A}$, define $\mathsf{sreqs}(x, \mathcal{A})$ as the set of subrequests $[\beta_0, \gamma_0] \rightsquigarrow [\beta_1, \gamma_1]...[\beta_k, \gamma_k]$ such that $[\beta_i, \gamma_i] \in \mathsf{pproj}_{\mathcal{A}}([b_i, c_i])$, for all $0 \leq i \leq k$. We refer to members of $\mathsf{sreqs}(x, \mathcal{A})$ as *secondary subrequests* of primary subrequest x and alignment $\mathcal{A}$. Figure 3b shows a secondary subrequest of the primary subrequest selected in Figure 3a.

Construction Requests: A *construction request* is a pair of subrequests of equivalent rank. Construction request $\langle x, \xi \rangle$ *matches* alignment $\mathcal{A}$ if $x \in \mathsf{preqs}(\mathcal{A})$ and $\xi \in \mathsf{sreqs}(x, \mathcal{A})$.

3.2 Construct the Unlabeled Rule

The basis of rule construction is the INDEXSORT operator, which takes as input a sequence of integers $I = \langle i_1, i_2, ..., i_k \rangle$, and an equivalent-length sequence of arbitrary values $\langle v_1, v_2, ..., v_k \rangle$, and returns a sequence $\langle v_{j_1}, v_{j_2}, ..., v_{j_k} \rangle$, where $\langle j_1, j_2, ..., j_k \rangle$ is a permutation of sequence I in ascending order. For instance, INDEXSORT$(\langle 4, 1, 50, 2 \rangle, \langle$"a", "b", "c", "d"$\rangle)$ = $\langle$"b", "d", "a", "c"$\rangle$. Note that the output of INDEXSORT(I, V) is nondeterministic if sequence I has repetitions. In Figure 4, we show the pseudocode for rule construction. We show an example construction in Figure 3 (bottom).

property is a property *of the alignment alone*. It is independent of the auxiliary information that GHKM uses, in particular the tree.

Primary Protocol RANKPP$_k$:

$$\{[b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_j, c_j] \text{ s.t. } 1 \leq b_0 \leq c_0 \text{ and } 0 \leq j \leq k\}$$

Primary Protocol MAPPP$_t$:

$$\{[b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_k, c_k] \text{ s.t. } \forall 0 \leq i \leq k\ [b_i, c_i] \in \mathsf{spans}(t)\}$$

Primary Protocol MAPPP$_T$:

$$\bigcup_{t \in T} \text{MAPPP}_t$$

Figure 5: Various primary protocols. Parameters: k is a nonnegative integer; t is a spanmap; T is a set of spanmaps (typically encoded as a forest).

3.3 Label the Rule

Rule construction produces unlabeled rules. To label these rules, we use a *labeling protocol*, defined as a function that takes a construction request as input, and returns a set of labels.

Figure 7 defines a number of general-purpose la-

```
**Secondary Protocol** TRIVSP$_{\mathcal{A}}(x)$:
    **return** $\mathsf{sreqs}(x, \mathcal{A})$

**Secondary Protocol** MAPSP$_{\tau,\mathcal{A}}(x)$:

    $\{[\beta_0, \gamma_0] \rightsquigarrow [\beta_1, \gamma_1]...[\beta_k, \gamma_k] \in \mathsf{sreqs}(x, \mathcal{A})$
      s.t. $\forall 0 \le i \le k : [\beta_i, \gamma_i] \in \mathsf{spans}(\tau)\}$
```

Figure 6: Various secondary protocols. Parameters: τ is a spanmap; $\mathcal{A}$ is an alignment; $x = [b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_k, c_k]$ is a subrequest.

beling protocols. Some of these are driven by trees. We will represent a tree as a *spanmap*, defined as a function that maps spans to symbol sequences. For instance, if a parse tree has constituent NP over span $[4, 7]$, then the corresponding spanmap t has $t([4, 7]) = \langle \text{NP} \rangle$. We map spans to *sequences* in order to accommodate unary chains in the parse tree. Nonconstituent spans are mapped to the empty sequence. For spanmap t, let $\mathsf{spans}(t)$ be the set of spans $[b, c]$ for which $t([b, c])$ is a nonempty sequence.

4 Extraction Programs

In the previous section, we developed a general technique for extracting labeled rules from aligned sentence pairs. Note that this was not an algorithm, but rather an algorithmic schema, as it left two questions unanswered:

1. What construction requests do we make?
2. What labeling protocol do we use?

We answer these questions with an *extraction program*, defined as a triple $\langle \mathcal{X}, \Xi, \mathcal{L} \rangle$, where:

- $\mathcal{X}$ is a set of subrequests, referred to as the *primary protocol*. It specifies the set of primary subrequests that interest us. Figure 5 defines some general-purpose primary protocols.
- Ξ maps every subrequest to a set of subrequests. We refer to Ξ as the *secondary protocol*. It specifies the set of secondary subrequests that interest us, given a particular primary subrequest. Figure 6 defines some general-purpose secondary protocols.
- $\mathcal{L}$ is a labeling protocol. Figure 7 defines some general-purpose labeling protocols.

```
**Labeling Protocol** TRIVLP$(x, \xi)$:
    **return** $\nabla^{k+1}$

**Labeling Protocol** PMAPLP$_t(x, \xi)$:

    $\{\langle l_0, ..., l_k \rangle$ s.t. $\forall 0 \le i \le k : l_i \in t([b_i, c_i])\}$

**Labeling Protocol** PMAPLP$_T(x, \xi)$:

    $\bigcup_{t \in T} \text{PMAPLP}_t(x, \xi)$

**Labeling Protocol** SMAPLP$_\tau(x, \xi)$:

    $\{\langle \lambda_0, ..., \lambda_k \rangle$ s.t. $\forall 0 \le i \le k : \lambda_i \in \tau([\beta_i, \gamma_i])\}$

**Labeling Protocol** SMAPLP$_{\mathcal{T}}(x, \xi)$:

    $\bigcup_{\tau \in \mathcal{T}} \text{SMAPLP}_\tau(x, \xi)$

**Labeling Protocol** IMAPLP$_{T,\mathcal{T}}(x, \xi)$:

    $\{\langle (l_0, \lambda_0), ..., (l_k, \lambda_k) \rangle$
      s.t. $\langle l_0, ..., l_k \rangle \in \text{PMAPLP}_T(x, \xi)$
      and $\langle \lambda_0, ..., \lambda_k \rangle \in \text{SMAPLP}_{\mathcal{T}}(x, \xi)\}$
```

Figure 7: Various labeling protocols. Parameters: t, τ are spanmaps; $T, \mathcal{T}$ are sets of spanmaps; $x = [b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_k, c_k]$ and $\xi = [\beta_0, \gamma_0] \rightsquigarrow [\beta_1, \gamma_1]...[\beta_k, \gamma_k]$ are subrequests.

Figure 8 shows the pseudocode for an "evaluator" that takes an extraction program (and an aligned sentence pair) as input and returns a set of labeled rules.

4.1 The GHKM Extraction Program

As previously stated, we developed our extraction schema by generalizing the GHKM algorithm (Galley et al., 2004). To recover GHKM as an instance of this schema, use the following program:

$$\text{EXTRACT}_{s,\sigma,\mathcal{A}}(\text{MAPPP}_t, \text{TRIVSP}_{\mathcal{A}}, \text{PMAPLP}_t)$$

where t is a spanmap encoding a parse tree over the primary sentence.

```
Algorithm EXTRACT$_{s,\sigma,\mathcal{A}}(\mathcal{X}, \Xi, \mathcal{L})$:
  $R = \{\}$
  for all subrequests $x \in \mathcal{X}$ do
    for all subrequests $\xi \in \Xi(x)$ do
      $U = \text{CONSTRUCTRULE}_{s,\sigma,\mathcal{A}}(x, \xi)$
      $L = \mathcal{L}(x, \xi)$
      $R = R \cup (U \times L)$
    end for
  end for
  return $R$
```

Figure 8: Evaluator for extraction programs. Parameters: $\langle s, \sigma, \mathcal{A}\rangle$ is an aligned sentence pair; $\mathcal{X}$ is a primary protocol; Ξ is a secondary protocol; $\mathcal{L}$ is a labeling protocol.

5 The Phrase Pair Extraction Program

In this section, we express phrase pair extraction (Och et al., 1999) as an extraction program.

For primary span $[b, c]$ and secondary span $[\beta, \gamma]$ of alignment $\mathcal{A}$, let $[b, c] \overset{\mathcal{A}}{\approx} [\beta, \gamma]$ if the following three conditions hold:

1. $a \overset{\mathcal{A}}{\sim} \alpha$ for some $a \in [b, c]$ and $\alpha \in [\beta, \gamma]$
2. $a \overset{\mathcal{A}}{\not\sim} \alpha$ for all $a \in [b, c]$ and $\alpha \notin [\beta, \gamma]$
3. $a \overset{\mathcal{A}}{\not\sim} \alpha$ for all $a \notin [b, c]$ and $\alpha \in [\beta, \gamma]$

Define the ruleset $\text{PBMT}(s, \sigma, \mathcal{A})$ to be the set of labeled rules $\langle r, \nabla^1\rangle$ such that:

- $r = \langle 0, \text{“}s_b...s_c\text{”}, \text{“}\sigma_\beta...\sigma_\gamma\text{”}, \emptyset\rangle$
- $[b, c] \overset{\mathcal{A}}{\approx} [\beta, \gamma]$

We want to express $\text{PBMT}(s, \sigma, \mathcal{A})$ as an extraction program. First we establish a useful lemma and corollary.

Lemma 1. $[b, c] \overset{\mathcal{A}}{\approx} [\beta, \gamma]$ *iff* $[b, c] \in \mathsf{frontier}(\mathcal{A})$ *and* $[\beta, \gamma] \in \mathsf{pproj}_{\mathcal{A}}([b, c])$.

Proof. Let $[b, c]^c = [1, b-1] \cup [c+1, |\mathcal{A}|_p]$.

$[b, c] \in \mathsf{frontier}(\mathcal{A})$ and $[\beta, \gamma] \in \mathsf{pproj}_{\mathcal{A}}([b, c])$

$$\overset{(1)}{\iff} \begin{cases} \mathsf{pmproj}_{\mathcal{A}}([b, c]) \cap \mathsf{pimage}_{\mathcal{A}}([b, c]^c) = \{\} \\ [\beta, \gamma] \in \mathsf{pproj}_{\mathcal{A}}([b, c]) \end{cases}$$

$$\overset{(2)}{\iff} \begin{cases} [\beta, \gamma] \cap \mathsf{pimage}_{\mathcal{A}}([b, c]^c) = \{\} \\ [\beta, \gamma] \in \mathsf{pproj}_{\mathcal{A}}([b, c]) \end{cases}$$

$$\overset{(3)}{\iff} \begin{cases} [\beta, \gamma] \cap \mathsf{pimage}_{\mathcal{A}}([b, c]^c) = \{\} \\ \mathsf{pimage}_{\mathcal{A}}([b, c]) \subseteq [\beta, \gamma] \end{cases}$$

$$\overset{(4)}{\iff} \begin{cases} \text{conditions 2 and 3 hold} \\ [\beta, \gamma] \neq \{\} \end{cases}$$

$$\overset{(5)}{\iff} \text{conditions 1, 2 and 3 hold}$$

Equivalence 1 holds by definition of $\mathsf{frontier}(\mathcal{A})$. Equivalence 2 holds because $[\beta, \gamma]$ differs from $\mathsf{pmproj}_{\mathcal{A}}([b, c])$ only in unaligned indices. Equivalence 3 holds because given the disjointness from $\mathsf{pimage}_{\mathcal{A}}([b, c]^c)$, $[\beta, \gamma]$ differs from $\mathsf{pimage}_{\mathcal{A}}([b, c])$ only in unaligned indices. Equivalences 4 and 5 are a restatement of conditions 2 and 3 plus the observation that empty spans can satisfy conditions 2 and 3. □

Corollary 2. *Consider monotonic subrequest* $x = [b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_k, c_k]$ *and arbitary subrequest* $\xi = [\beta_0, \gamma_0] \rightsquigarrow [\beta_1, \gamma_1]...[\beta_k, \gamma_k]$. *Construction request* $\langle x, \xi\rangle$ *matches alignment* $\mathcal{A}$ *iff* $[b_i, c_i] \overset{\mathcal{A}}{\approx} [\beta_i, \gamma_i]$ *for all* $0 \le i \le k$.

We are now ready to express the rule set $\text{PBMT}(s, \sigma, \mathcal{A})$ as an extraction program.

Theorem 3. $\text{PBMT}(s, \sigma, \mathcal{A}) = \text{EXTRACT}_{s,\sigma,\mathcal{A}}(\text{RANKPP}_0, \text{TRIVSP}_{\mathcal{A}}, \text{TRIVLP})$

Proof.

$\langle r, l\rangle \in \text{EXT}_{s,\sigma,\mathcal{A}}(\text{RANKPP}_0, \text{TRIVSP}_{\mathcal{A}}, \text{TRIVLP})$

$$\overset{(1)}{\iff} \begin{cases} x = [b, c] \rightsquigarrow \epsilon \text{ and } \xi = [\beta, \gamma] \rightsquigarrow \epsilon \\ \langle x, \xi\rangle \text{ matches alignment } \mathcal{A} \\ \{r\} = \text{CONSTRUCTRULE}_{s,\sigma,\mathcal{A}}(x, \xi) \\ l = \nabla^1 \end{cases}$$

$$\overset{(2)}{\iff} \begin{cases} x = [b, c] \rightsquigarrow \epsilon \text{ and } \xi = [\beta, \gamma] \rightsquigarrow \epsilon \\ \langle x, \xi\rangle \text{ matches alignment } \mathcal{A} \\ r = \langle 0, \text{“}s_b...s_c\text{”}, \text{“}\sigma_\beta...\sigma_\gamma\text{”}, \emptyset\rangle \\ l = \nabla^1 \end{cases}$$

$$\overset{(3)}{\iff} \begin{cases} [b, c] \overset{\mathcal{A}}{\approx} [\beta, \gamma] \\ r = \langle 0, \text{“}s_b...s_c\text{”}, \text{“}\sigma_\beta...\sigma_\gamma\text{”}, \emptyset\rangle \\ l = \nabla^1 \end{cases}$$

$$\overset{(4)}{\iff} \langle r, l\rangle \in \text{PBMT}(s, \sigma, \mathcal{A})$$

Equivalence 1 holds by the definition of EXTRACT and RANKPP$_0$. Equivalence 2 holds by the pseudocode of CONSTRUCTRULE. Equivalence 3 holds from Corollary 2. Equivalence 4 holds from the definition of PBMT$(s, \sigma, \mathcal{A})$. □

6 The Hiero Extraction Program

In this section, we express the hierarchical phrase-based extraction technique of (Chiang, 2007) as an extraction program. Define HIERO$_0(s, \sigma, \mathcal{A})$ = PBMT$(s, \sigma, \mathcal{A})$. For positive integer k, define HIERO$_k(s, \sigma, \mathcal{A})$ as the smallest superset of HIERO$_{k-1}(s, \sigma, \mathcal{A})$ satisfying the following condition:

- For any labeled rule $\langle\langle k-1, s^*, \sigma^*, \pi\rangle, \nabla^k\rangle \in$ HIERO$_{k-1}(s, \sigma, \mathcal{A})$ such that:
  1. $s^* = s_1^* \cdot \text{“}s_b ... s_c\text{”} \cdot s_2^*$
  2. $\sigma^* = \sigma_1^* \cdot \text{“}\sigma_\beta ... \sigma_\gamma\text{”} \cdot \sigma_2^*$
  3. $\pi = \langle \pi_1, \pi_2, ..., \pi_{k-1}\rangle$
  4. s_2^* has rank 0.[4]
  5. σ_1^* has rank j.
  6. $[b, c] \overset{\mathcal{A}}{\sim} [\beta, \gamma]$

  it holds that labeled rule $\langle r, \nabla^{k+1}\rangle$ is a member of HIERO$_k(s, \sigma, \mathcal{A})$, where r is:

$$\langle k, s_1^* \cdot \text{“}\nabla\text{”} \cdot s_2^*, \sigma_1^* \cdot \text{“}\nabla\text{”} \cdot \sigma_2^*, \langle \pi_1, ..., \pi_j, k, \pi_{j+1}, ..., \pi_{k-1}\rangle\rangle$$

Theorem 4. HIERO$_k(s, \sigma, \mathcal{A})$ = EXTRACT$_{s,\sigma,\mathcal{A}}$(RANKPP$_k$, TRIVSP$_\mathcal{A}$, TRIVLP)

Proof. By induction. Define $\mathsf{ext}(k)$ to mean EXTRACT$_{s,\sigma,\mathcal{A}}$(RANKPP$_k$, TRIVSP$_\mathcal{A}$, TRIVLP). From Theorem 3, HIERO$_0(s, \sigma, \mathcal{A}) = \mathsf{ext}(0)$. Assume that HIERO$_{k-1}(s, \sigma, \mathcal{A}) = \mathsf{ext}(k-1)$ and prove that HIERO$_k(s, \sigma, \mathcal{A}) \backslash$HIERO$_{k-1}(s, \sigma, \mathcal{A}) = \mathsf{ext}(k)\backslash\mathsf{ext}(k-1)$.

$$\langle r', l'\rangle \in \mathsf{ext}(k)\backslash\mathsf{ext}(k-1)$$

$$\overset{(1)}{\iff} \begin{cases} x' = [b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_k, c_k] \\ \xi' = [\beta_0, \gamma_0] \rightsquigarrow [\beta_1, \gamma_1]...[\beta_k, \gamma_k] \\ \langle x', \xi'\rangle \text{ matches alignment } \mathcal{A} \\ \{r'\} = \text{CONSTRUCTRULE}_{s,\sigma,\mathcal{A}}(x', \xi') \\ l' = \nabla^{k+1} \end{cases}$$

$$\overset{(2)}{\iff} \begin{cases} x = [b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_{k-1}, c_{k-1}] \\ \xi = [\beta_0, \gamma_0] \rightsquigarrow [\beta_1, \gamma_1]...[\beta_{k-1}, \gamma_{k-1}] \\ \{r\} = \text{CONSTRUCTRULE}_{s,\sigma,\mathcal{A}}(x, \xi) \\ \pi = \langle \pi_1, ..., \pi_{k-1}\rangle \\ r = \begin{array}{l}\langle k-1, s_1^* \cdot \text{“}s_{b_k} ... s_{c_k}\text{”} \cdot s_2^*, \\ \qquad \sigma_1^* \cdot \text{“}\sigma_{\beta_k} ... \sigma_{\gamma_k}\text{”} \cdot \sigma_2^*, \pi\rangle\end{array} \\ s_2^* \text{ has rank 0 and } \sigma_1^* \text{ has rank } j \\ x' = [b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_k, c_k] \\ \xi' = [\beta_0, \gamma_0] \rightsquigarrow [\beta_1, \gamma_1]...[\beta_k, \gamma_k] \\ \langle x', \xi'\rangle \text{ matches alignment } \mathcal{A} \\ \pi' = \langle \pi_1, ..., \pi_j, k, \pi_{j+1}, ..., \pi_{k-1}\rangle \\ r' = \langle k, s_1^* \cdot \text{“}\nabla\text{”} \cdot s_2^*, \sigma_1^* \cdot \text{“}\nabla\text{”} \cdot \sigma_2^*, \pi'\rangle \\ l' = \nabla^{k+1} \end{cases}$$

$$\overset{(3)}{\iff} \begin{cases} \pi = \langle \pi_1, ..., \pi_{k-1}\rangle \\ r = \begin{array}{l}\langle k-1, s_1^* \cdot \text{“}s_{b_k} ... s_{c_k}\text{”} \cdot s_2^*, \\ \qquad \sigma_1^* \cdot \text{“}\sigma_{\beta_k} ... \sigma_{\gamma_k}\text{”} \cdot \sigma_2^*, \pi\rangle\end{array} \\ s_2^* \text{ has rank 0 and } \sigma_1^* \text{ has rank } j \\ \langle r, \nabla^k\rangle \in \text{HIERO}_{k-1}(s, \sigma, \mathcal{A}) \\ \pi' = \langle \pi_1, ..., \pi_j, k, \pi_{j+1}, ..., \pi_{k-1}\rangle \\ r' = \langle k, s_1^* \cdot \text{“}\nabla\text{”} \cdot s_2^*, \sigma_1^* \cdot \text{“}\nabla\text{”} \cdot \sigma_2^*, \pi'\rangle \\ [b_i, c_i] \overset{\mathcal{A}}{\sim} [\beta_i, \gamma_i] \text{ for all } 0 \le i \le k \\ l' = \nabla^{k+1} \end{cases}$$

$$\overset{(4)}{\iff} \langle r', l'\rangle \in \text{HIERO}_k(s, \sigma, \mathcal{A})\backslash\text{HIERO}_{k-1}(s, \sigma, \mathcal{A})$$

Equivalence 1 holds by the definition of $\mathsf{ext}(k)\backslash\mathsf{ext}(k-1)$. Equivalence 2 holds by the pseudocode of CONSTRUCTRULE. Equivalence 3 holds by the inductive hypothesis and Corollary 2. Equivalence 4 holds by the definition of HIERO$_k(s, \sigma, \mathcal{A})\backslash$HIERO$_{k-1}(s, \sigma, \mathcal{A})$. □

7 Discussion

In this paper, we have created a framework that allows us to express a desired rule extraction method as a set of construction requests and a labeling protocol. This enables a modular, “mix-and-match” approach to rule extraction. In Table 1, we summarize the results of this paper, as well as our conjectured extraction programs for several other methods. For instance, Syntax-Augmented Machine Translation (SAMT) (Zollmann and Venugopal, 2006) is a

[4]This condition is not in the original definition. It is a cosmetic addition, to enforce the consecutive ordering of variable indices on the rule LHS.

hybridization of Hiero and GHKM that uses the primary protocol of Hiero and the labeling protocol of GHKM. To bridge the approaches, SAMT employs a fuzzy version[5] of the spanmap t that assigns a trivial label to non-constituent primary spans:

$$\tilde{t}([b,c]) = \begin{cases} t([b,c]) & \text{if } [b,c] \in \mathsf{spans}(t) \\ \langle \nabla \rangle & \text{otherwise} \end{cases}$$

Other approaches can be similarly expressed as straightforward variants of the extraction programs we have developed in this paper.

Although we have focused on idealized methods, this framework also allows a compact and precise characterization of practical restrictions of these techniques. For instance, (Chiang, 2007) lists six criteria that he uses in practice to restrict the generation of Hiero rules. His condition 4 ("Rules can have at most two nonterminals.") and condition 5 ("It is prohibited for nonterminals to be adjacent on the French side.") can be jointly captured by replacing Hiero's primary protocol with the following:

$$\{[b_0, c_0] \rightsquigarrow [b_1, c_1]...[b_j, c_j] \text{ s.t. } \begin{array}{l} 1 \leq b_0 \leq c_0 \\ 0 \leq j \leq \mathbf{2} \\ \mathbf{b_2 > c_1 + 1}\} \end{array}$$

His other conditions can be similarly captured with appropriate changes to Hiero's primary and secondary protocols.

This work is primarily intended as a survey of the dominant translation rule extraction paradigms, in which we make explicit the close relationship between these approaches, and establish a language for future hybridizations. From a practical perspective, we facilitate a generic and extensible implementation which supports a wide variety of existing methods, and which permits the precise expression of practical extraction heuristics.

[5]This corresponds with the original formulation of Syntax Augmented Machine Translation (Zollmann and Venugopal, 2006). More recent versions of SAMT adopt a more refined "fuzzifier" that assigns hybrid labels to non-constituent primary spans.

References

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.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of ACL*, pages 1443–1452.

A. de Gispert, G. Iglesias, G. Blackwood, E.R. Banga, and W. Byrne. 2010. Hierarchical phrase-based translation with weighted finite state transducers and shallow-n grammars. *Computational Linguistics*, 36(3):505–533.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of ACL*, pages 205–208.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic models. In *Proceedings of ACL-COLING*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth ACL Workshop on Statistical Machine Translation*, pages 135–139.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of ACL/COLING*, pages 609–616.

Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *Proceedings of ACL*.

Yang Liu, Yajuan Lu, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proceedings of ACL/IJCNLP*, pages 558–566.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of EMNLP-CoNLL*.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. Spmt: Statistical machine translation with syntactified target language phrases. In *Proceedings of EMNLP*, pages 44–52.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP*.

Franz J. Och, Christof Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the Joint Conf. of Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

Ashish Venugopal, Andreas Zollmann, and Stephan Vogel. 2007. An efficient two-pass approach to synchronous-cfg driven statistical mt. In *Proceedings of HLT/NAACL*.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of ACL*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of NAACL Workshop on Statistical Machine Translation*.