Phrase Dependency Machine Translation with Quasi-Synchronous Tree-to-Tree Features

Kevin Gimpel*
Toyota Technological Institute
at Chicago

Noah A. Smith**
Carnegie Mellon University

Recent research has shown clear improvement in translation quality by exploiting linguistic syntax for either the source or target language. However, when using syntax for both languages ("tree-to-tree" translation), there is evidence that syntactic divergence can hamper the extraction of useful rules (Ding and Palmer 2005). Smith and Eisner (2006) introduced quasi-synchronous grammar, a formalism that treats non-isomorphic structure softly using features rather than hard constraints. Although a natural fit for translation modeling, its flexibility has proved challenging for building real-world systems. In this article, we present a tree-to-tree machine translation system inspired by quasi-synchronous grammar. The core of our approach is a new model that combines phrases and dependency syntax, integrating the advantages of phrase-based and syntax-based translation. We report statistically significant improvements over a phrase-based baseline on five of seven test sets across four language pairs. We also present encouraging preliminary results on the use of unsupervised dependency parsing for syntax-based machine translation.

1. Introduction

Building translation systems for many language pairs requires addressing a wide range of translation divergence phenomena. Several researchers have studied divergence between languages in corpora and found it to be considerable, even for closely related languages (Dorr 1994; Fox 2002; Wellington, Waxmonsky, and Melamed 2006; Søgaard and Kuhn 2009). To address this, many have incorporated **linguistic syntax** into translation model design. The statistical natural language processing (NLP) community has developed automatic parsers that can produce syntactic analyses for sentences in several languages (Klein and Manning 2003; Buchholz and Marsi 2006; Nivre et al.

* Toyota Technological Institute at Chicago, Chicago, IL 60637. E-mail: kgimpel@ttic.edu.
** School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.
E-mail: nasmith@cs.cmu.edu.

Submission received: 10 November 2012; revised submission received: 12 May 2013; accepted for publication: 23 June 2013.

doi:10.1162/COLI_a_00175

2007). The availability of these parsers, and gains in their accuracy, triggered research interest in **syntax-based** statistical machine translation (Yamada and Knight 2001).

Syntax-based translation models are diverse, using different grammatical formalisms and features. Some use a parse tree for the source sentence ("tree-to-string"), others produce a parse when generating the target sentence ("string-to-tree"), and others combine both ("tree-to-tree"). We focus on the final category in this article. Tree-to-tree translation has proved to be a difficult modeling problem, as initial attempts at it underperformed systems that used no syntax at all (Cowan, Kučerová, and Collins 2006; Ambati and Lavie 2008; Liu, Lü, and Liu 2009). Subsequent research showed that substantial performance gains can be achieved if hard constraints—specifically, isomorphism between a source sentence's parse and the parse of its translation—are relaxed (Liu, Lü, and Liu 2009; Chiang 2010; Zhang, Zhai, and Zong 2011; Hanneman and Lavie 2011). This suggests that constraints must be handled with care.

Yet the classic approach to tree-to-tree translation imposes hard constraints through the use of **synchronous grammars** developed for programming language compilation (Aho and Ullman 1969). A synchronous grammar derives two strings simultaneously: one in the source language and one in the target language. A single derivation is used for both strings, which limits the divergence phenomena that can be captured. As a result, researchers have developed synchronous grammars with larger rules that, rule-internally, capture more phenomena, typically at increased computational expense (Shieber and Schabes 1990; Eisner 2003; Gildea 2003; Ding and Palmer 2005).

We take a different approach. We take inspiration from a family of formalisms called **quasi-synchronous grammar** (QG; Smith and Eisner 2006). Unlike synchronous grammar, QG assumes the entire input sentence and some syntactic parse of it are provided and fixed. QG then defines a *monolingual* grammar whose language is a set of translations inspired by the input sentence and tree. The productions in this monolingual grammar generate a piece of the translation's tree and align it to a piece of the fixed input tree. Therefore, arbitrary non-isomorphic structures are possible between the two trees. A weighted QG uses feature functions to softly penalize or encourage particular types of syntactic divergence.

In this article, we present a statistical tree-to-tree machine translation system inspired by quasi-synchronous grammar. We exploit the flexibility of QG to develop a new syntactic translation model that seeks to combine the benefits of both phrase-based and syntax-based translation. Our model organizes phrases into a tree structure inspired by dependency syntax (Tesnière 1959). Instead of standard dependency trees in which *words* are vertices, our trees have *phrases* as vertices. The result captures phenomena like local reordering and idiomatic translations within phrases, as well as long-distance relationships among the phrases in a sentence. We use the term **phrase dependency tree** when referring to this type of dependency tree; phrase dependencies have also been used by Wu et al. (2009) for opinion mining and previously for machine translation by Hunter and Resnik (2010). Because we combine phrase dependencies with features from quasi-synchronous grammar, we refer to our model as a **quasi-synchronous phrase dependency** (QPD) translation model.

Our tree-to-tree approach requires parsers for both the source and target languages. For two of the language pairs we consider (Chinese→English and German→English), treebanks of hand-annotated parse trees are available (e.g., the Penn Treebank; Marcus, Santorini, & Marcinkiewitz 1993), allowing the use of highly accurate statistical parsers (Levy and Manning 2003; Rafferty and Manning 2008; Martins, Smith, and Xing 2009). We also want to apply our model to languages that do not have tree-

banks (e.g., Urdu and Malagasy), and for this we turn to **unsupervised parsing**. The NLP community has developed a range of statistical algorithms for building unsupervised parsers (Klein and Manning 2002, 2004; Smith 2006; Blunsom and Cohn 2010; Naseem et al. 2010; Spitkovsky, Alshawi, and Jurafsky 2010; Cohen 2011). They require only raw, unannotated text in the language of interest, making them ideal for use in translation.

Unsupervised *shallow* syntactic analysis has been used successfully for translation modeling by Zollmann and Vogel (2011), who showed that unsupervised part-of-speech tags could be used to label the hierarchical translation rules of Chiang (2005) to match the performance of a system that uses supervised full syntactic parses. We take additional steps in this direction, leveraging state-of-the-art unsupervised models for full syntactic analysis (Klein and Manning 2004; Berg-Kirkpatrick et al. 2010; Gimpel and Smith 2012a) to obtain improvements in translation quality. We find that replacing a supervised parser for Chinese with an unsupervised one has no effect on performance, and using an unsupervised English parser only hurts slightly. We use unsupervised parsing to apply our full model to Urdu→English and English→Malagasy translation, reporting statistically significant improvements over our baselines. These initial results offer promise for researchers to apply syntactic translation models to the thousands of languages for which we do not have manually annotated corpora, and naturally suggest future research directions.

The rest of this article is laid out as follows. In Section 2, we discuss quasi-synchronous grammar and dependency syntax and motivate our modeling choices. We present our translation model in Section 3, describe how we extract rules in Section 4, and list our feature functions in Section 5. Decoding algorithms are given in Section 6. We present experiments measuring our system's performance on translation tasks involving four language pairs and several test sets in Section 7. We find statistically significant improvements over a strong phrase-based baseline on five out of seven test sets across four language pairs. We also perform a human evaluation to study how our system improves translation quality. This article is a significantly expanded version of Gimpel and Smith (2011), containing additional features, a new decoding algorithm, and a more thorough experimental evaluation. It presents key material from Gimpel (2012), to which readers seeking further details are referred.

2. Background and Motivation

We begin by laying groundwork for the rest of the article. We define notation in Section 2.1. Section 2.2 discusses how synchronous and quasi-synchronous grammar handle syntactic divergence. In Section 2.3, we introduce dependency syntax and review prior work that has used it for machine translation. Section 2.4 presents two examples of syntactic divergence that motivate the model we develop in Section 3.

2.1 Notation

We use boldface for vectors and we denote individual elements in vectors using subscripts; for example, the source and target sentences are denoted $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ and $\boldsymbol{y} = \langle y_1, \ldots, y_m \rangle$. We denote sequences of elements in vectors using subscripts and superscripts; for example, the sequence from source word i to source word j (inclusive) is denoted $\boldsymbol{x}_i^j$, and therefore $\boldsymbol{x}_i^i = \langle x_i \rangle$. We denote the set containing the first k positive integers as $[k]$. This notation is summarized in Table 1.

Table 1
Notation used in this article.

i, j, k, l	integers
$\boldsymbol{x}, \boldsymbol{y}$	vectors
x_i	entry i in vector $\boldsymbol{x}$
$\boldsymbol{x}_i^j$	sequence from entry i to entry j (inclusive) in vector $\boldsymbol{x}$
$[i]$	the set containing the first i positive integers
$\lvert\boldsymbol{x}\rvert$	length of vector $\boldsymbol{x}$

2.2 Synchronous and Quasi-Synchronous Grammars

To model syntactic transformations, researchers have developed powerful grammatical formalisms, many of which are variations of **synchronous grammars**. The most widely used is synchronous context-free grammar (Wu 1997; Gildea 2003; Chiang 2005; Melamed 2003), an extension of context-free grammar to a bilingual setting where two strings are generated simultaneously with a single derivation. Synchronous context-free grammars are computationally attractive but researchers have shown that they cannot handle certain phenomena in manually aligned parallel data (Wellington, Waxmonsky, and Melamed 2006; Søgaard and Kuhn 2009). Figure 1 shows two such examples of word alignment patterns in German–English data. These patterns were called "cross-serial discontinuous translation units" (CDTUs) by Søgaard and Kuhn (2009). CDTUs cannot even be handled by the more sophisticated synchronous formalisms given by Eisner (2003) and Ding and Palmer (2005). CDTUs *can* be handled by synchronous tree adjoining grammar (STAG; Shieber and Schabes 1990), but STAG comes with substantially heftier computational requirements. Furthermore, Søgaard and Kuhn (2009) found examples in parallel data that even STAG cannot handle.

Smith and Eisner (2006) noted that these limitations of synchronous grammars result from an emphasis on *generating* the two strings. However, for many real-world applications, such as translation, one of the sentences is provided. The model only needs to score translations of the given source sentence, not provide a generative account for sentence *pairs*. Smith and Eisner proposed an alternative to synchronous grammar—**quasi-synchronous grammar** (QG)—that exploits this fact for increased flexibility in translation modeling. A QG assumes the source sentence and a parse are given and scores possible translations of the source sentence along with their parses. That is, a quasi-synchronous grammar is a *monolingual* grammar that derives strings in the target language. The strings' derivations are scored using feature functions on an alignment from nodes in the target tree to nodes in the source tree. The quasi-synchronous dependency grammars of Smith and Eisner (2006) and Gimpel and Smith (2009b) can generate the translations in Figure 1, as can phrase-based models like Moses (Koehn et al. 2007) and the phrase dependency model we present in Section 3.

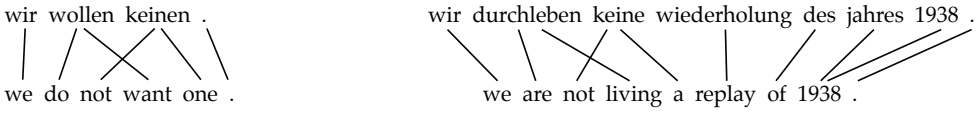


Figure 1
Examples of word alignment patterns in German–English that require the increased expressive power of synchronous tree adjoining grammar.

Quasi-synchronous grammar, like synchronous grammar, can in principle be instantiated for a wide range of formalisms. Dependency syntax (which we discuss in Section 2.3) has been used in most previous applications of QG, including word alignment (Smith and Eisner 2006) and machine translation (Gimpel and Smith 2009b). Aside from translation, QG has been used for a variety of applications involving relationships among sentences, including question answering (Wang, Smith, and Mitamura 2007), paraphrase identification (Das and Smith 2009), parser projection and adaptation (Smith and Eisner 2009), title generation (Woodsend, Feng, and Lapata 2010), sentence simplification (Woodsend and Lapata 2011), information retrieval (Park, Croft, and Smith 2011), and supervised parsing from multiple treebanks with different annotation conventions (Li, Liu, and Che 2012).

2.3 Dependency Syntax and Machine Translation

Many syntactic theories have been applied to translation modeling, but we focus in this article on dependency syntax (Tesnière 1959). Dependency syntax is a lightweight formalism that builds trees consisting of a set of directed arcs from words to their syntactic heads (also called "parents"). Examples of dependency trees are shown in Figure 2. Each word has exactly one parent, and \$ is a special "wall" symbol that is located at position 0 in the sentence and acts as parent to words that have no other parent in the sentence. Formally, a **dependency tree** on an m-word sentence $\boldsymbol{y}$ is a function $\tau_{\boldsymbol{y}} : \{1, \ldots, m\} \rightarrow \{0, \ldots, m\}$ where $\tau_{\boldsymbol{y}}(i)$ is the index of the parent of word y_i. If $\tau_{\boldsymbol{y}}(i) = 0$, we say word y_i is a **root** of the tree. The function $\tau_{\boldsymbol{y}}$ is not permitted to have cycles. We restrict our attention to **projective** dependency trees in this article. Projective dependency trees are informally defined as having no crossing arcs when all dependencies are drawn on one side of the sentence. See Kübler, McDonald, and Nivre (2009) for formal definitions of these terms.

Researchers have shown that dependency trees are better preserved when projecting across word alignments than phrase structure trees (Fox 2002). This makes dependency syntax appealing for translation modeling, but to date there are not many tree-to-tree translation models that use dependency syntax on both sides. One exception is the system of Ding and Palmer (2005), who used a synchronous tree substitution grammar designed for dependency syntax, capturing non-isomorphic structure within rules using elementary trees. Another is the system of Riezler and Maxwell III (2006), who used lexical-functional dependency trees on both sides and also include phrase translation rules. Relatedly, Quirk, Menezes, and Cherry (2005) used a source-side dependency parser and projected automatic parses across word alignments in order to model dependency syntax on phrase pairs.

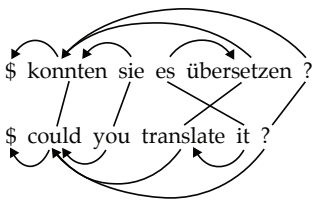


Figure 2
Examples of dependency trees with word alignment. Arrows are drawn from children to parents. A child word is a modifier of its parent. Each word has exactly one parent and \$ is a special "wall" symbol that serves as the parent of all **root** words in the tree (i.e., those with no other parent).

But most who have used dependency syntax have done so either on the source side in tree-to-string systems (Lin 2004; Xiong, Liu, and Lin 2007; Xie, Mi, and Liu 2011) or the target side in string-to-tree systems (Shen, Xu, and Weischedel 2008; Carreras and Collins 2009; Galley and Manning 2009; Hunter and Resnik 2010; Su et al. 2010; Tu et al. 2010). Others have added features derived from source dependency parses to phrase-based or hierarchical phrase-based translation models (Gimpel and Smith 2008; Gao, Koehn, and Birch 2011).

2.4 Motivating Examples

Although Fox (2002) found that dependencies are more often preserved across hand-aligned bitext than constituents, there are still several concerns when using dependency syntax for tree-to-tree translation. First, we only have hand-aligned sentence pairs for small data sets and few language pairs, so in practice we must deal with the noise in automatic word aligners and parsers. Second, not all dependencies are preserved in hand-aligned data, so we would need to be able to handle non-isomorphic structure even if we did have perfect tools. The model we present in Section 3 avoids isomorphism constraints from synchronous grammar and encourages dependency preservation across languages by using dependencies on **phrases**—flat multi-word units—rather than words.

To motivate these choices, we now give two frequently occurring examples of dependency tree-to-tree divergence in German–English data.[1] We consider the German–English parallel corpus used in our experiments (and described in Appendix A). We parsed the English side using TurboParser (Martins et al. 2010), a state-of-the-art dependency parser. TurboParser was trained on the Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993) converted to dependencies using the Yamada-Matsumoto head rules (Yamada and Matsumoto 2003). We parsed the German side using the factored model in the Stanford parser (Rafferty and Manning 2008), which is trained from the NEGRA phrase-structure treebank (Skut et al. 1997). The Stanford parser's source code defines a set of head rules for converting the phrase-structure parse output to dependencies.[2]

The first example is shown in Figure 3. The bold words illustrate a "sibling" relationship, meaning that the source words aligned to the parent and child in the English sentence have the same parent on the German side. Many sibling configurations appear when the English dependency is DET→N within a PP. By convention, the NEGRA treebank uses flat structures for PPs like "P DET N" rather than using a separate NP for DET N. When the parser converts this to a dependency tree, the DET and N are made children of the P. In English dependency parsing, due to the Penn Treebank conventions, the DET is made a child of the N, which is a child of the P. There are many other instances like this one that frequently lie within PPs, like *the→us* and *recent→years*. However, if we tokenized *the us* as a phrase and also *den usa*, then both would be children of the preposition, and the dependency would be preserved.

1 The study that uncovered these examples is detailed in Gimpel (2012). It gives evidence of frequent non-isomorphic dependency structure between German and English with automatic word aligners and parsers.

2 These rules use comparable conventions to the Yamada-Matsumoto head rules for English (modulo the differences in languages and tag/label sets): finite verbs are sentence roots, adpositions are heads of adpositional phrases, nouns are heads of noun phrases, and so forth.

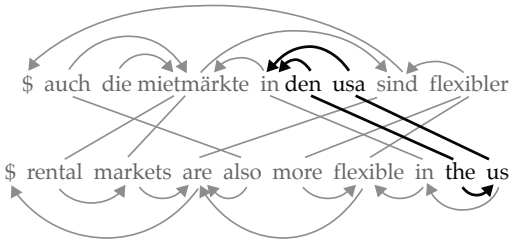


Figure 3
Example of a sentence pair containing a frequently-observed "sibling" relationship in German–English data: in the *the→us* dependency, the aligned German words are siblings in the source dependency tree. This occurs due to differences in treebank and head rule conventions between the two data sets. The German parser produces flat PPs with little internal structure, so when the dependency tree is generated, each word in the PP attaches to the P, the head of the phrase.

The second example is shown in Figure 4, which gives an example of a "grandparent-grandchild" relationship. In the English dependency *until←recently*, the aligned source words are in a grandparent relationship in the source sentence's dependency tree. We note, however, that if *vor kurzem* is tokenized as a phrase, then we might let the entire phrase be the child of *bis*, preserving the dependency across languages.

By considering phrasal structure and dependencies among *phrases*, we can reduce some of the syntactic divergence in real-world data. The model we develop in the next section is based on this idea.

3. Model

In the previous section we noted two examples in which flattening dependency tree structure into "phrasal dependencies" could improve dependency preservation between German and English. This idea is compatible with the well-known principle that translation quality is improved when larger units are modeled within translation rules. For example, improvements were found by moving from word-based models to so-called **phrase-based** translation models. Modern phrase-based translation systems are typified by the Moses system (Koehn et al. 2007), based on the approach presented by Koehn, Och, and Marcu (2003). Phrase-based models excel at capturing local reordering phenomena and memorizing multi-word translations.

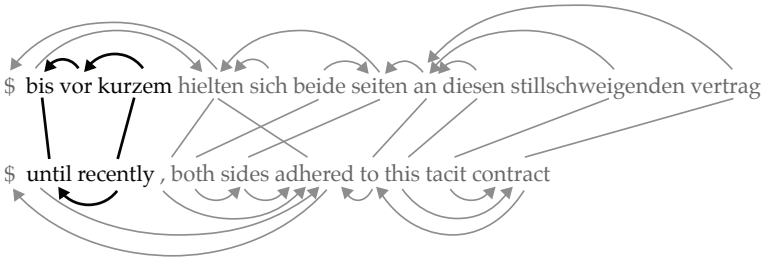


Figure 4
Example of a sentence pair containing a frequently-observed "grandparent-grandchild" relationship in German–English data: the English parent and child words in the *until←recently* dependency are aligned to German words in a grandparent-grandchild relationship.

On the other hand, models that use rules employing syntax (Yamada and Knight 2001) or syntax-like representations (Chiang 2005) handle long-distance reordering better than phrase-based systems (Birch, Blunsom, and Osborne 2009), and therefore perform better for certain language pairs (Zollmann et al. 2008). In order to better handle syntactic divergence and obtain the benefits of these two types of models, we use rules that combine phrases and syntax. In particular, our rules use dependencies between phrases rather than words; we call them **phrase dependencies**. When adding in source syntax, we eschew the constraints of synchronous grammar in favor of the feature-based approach of quasi-synchronous grammar. So we call our model a **quasi-synchronous phrase dependency** (QPD) translation model.

In Section 3.1, we define phrase dependency trees and in Section 3.2 we present our model. We discuss rule extraction in Section 4 and define the feature functions in the model in Section 5. Decoding is discussed in Section 6 and an empirical evaluation is given in Section 7. Key definitions used throughout this section and the remaining sections are listed in Table 2.

3.1 Phrase Dependencies

In Section 2.3 we defined dependency trees. Now we provide an analogous definition for *phrase* dependency trees. We first define a segmentation of a sentence into phrases. Given a sentence $\boldsymbol{y}$, where $m = |\boldsymbol{y}|$, we define a **phrase** ϕ as a word sequence $\boldsymbol{y}_j^k$, for j and

Table 2
Key definitions for our model.

$\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$	source language sentence
$\boldsymbol{y} = \langle y_1, \ldots, y_m \rangle$	target language sentence, translation of $\boldsymbol{x}$
$\pi = \boldsymbol{x}_j^k$	source-sentence phrase: subsequence of words in the source sentence $\boldsymbol{x}$, i.e., $1 \leq j \leq k \leq n$; the number of words in π is $\lvert\pi\rvert$
$\phi = \boldsymbol{y}_j^k$	target-sentence phrase: subsequence of words in the target sentence $\boldsymbol{y}$, i.e., $1 \leq j \leq k \leq m$
$\pi = \langle \pi_1, \ldots, \pi_{n'} \rangle$	segmentation of $\boldsymbol{x}$ into phrases such that for $i \in [n']$, $\pi_i = \boldsymbol{x}_j^k$ is a source-sentence phrase and $\pi_1 \cdot \ldots \cdot \pi_{n'} = \boldsymbol{x}$
$\phi = \langle \phi_1, \ldots, \phi_{n'} \rangle$	segmentation of $\boldsymbol{y}$ into phrases such that for $i \in [n']$, $\phi_i = \boldsymbol{y}_j^k$ is a target-sentence phrase and $\phi_1 \cdot \ldots \cdot \phi_{n'} = \boldsymbol{y}$
$\boldsymbol{b} : \{1, \ldots, n'\} \to \{1, \ldots, n'\}$	one-to-one alignment (bijection) from phrases in ϕ to phrases in π; for all $i \in [n']$, if $\boldsymbol{b}(i) = j$, then π_j is a subsequence of $\boldsymbol{x}$ and ϕ_i is a subsequence of $\boldsymbol{y}$
$\tau_{\boldsymbol{x}} : \{1, \ldots, n\} \to \{0, \ldots, n\}$	dependency tree on source words $\boldsymbol{x}$, where $\tau_{\boldsymbol{x}}(i)$ is the index of the parent of word $\boldsymbol{x}_i$ (0 is the wall symbol \$)
$\tau_\phi : \{1, \ldots, n'\} \to \{0, \ldots, n'\}$	dependency tree on target phrases ϕ, where $\tau_\phi(i)$ is the index of the parent of phrase ϕ_i (0 is the wall symbol \$)
$\boldsymbol{h} = \langle \boldsymbol{h}', \boldsymbol{h}'' \rangle$	vector of feature functions; $\boldsymbol{h}'$ holds the Moses feature functions and $\boldsymbol{h}''$ holds the QPD feature functions
$\boldsymbol{\theta} = \langle \boldsymbol{\theta}', \boldsymbol{\theta}'' \rangle$	vector of feature weights for $\boldsymbol{h}$

k such that $1 \leq j \leq k \leq m$. The number of words in phrase ϕ is denoted $|\phi|$. We define a **phrase segmentation** of $\boldsymbol{y}$ as $\phi = \langle \phi_1, \ldots, \phi_{n'} \rangle$ such that for $i \in [n']$, $\phi_i = \boldsymbol{y}_j^k$ is a phrase and $\phi_1 \cdot \ldots \cdot \phi_{n'} = \boldsymbol{y}$, where $\cdot$ denotes string concatenation.

Given a phrase segmentation ϕ, we define a **phrase dependency tree** as a function $\tau_\phi : [n'] \rightarrow \{0\} \cup [n']$ where $\tau_\phi(i)$ is the index of the parent of phrase ϕ_i. If $\tau_\phi(i) = 0$, we say phrase ϕ_i is the **root** of the phrase dependency tree; we require there to be exactly one root phrase. As with dependency trees, τ_ϕ cannot have cycles.[3] To distinguish phrase dependency trees from the ordinary dependency trees defined in Section 2.3, we will sometimes refer to the latter as "lexical dependency trees."

Phrase dependency trees have also been used by Wu et al. (2009) to extract features for opinion mining and a similar formalism was used previously for machine translation by Hunter and Resnik (2010). Phrase dependencies allow us to capture phenomena like local reordering and idiomatic translations within each phrase as well as longer-distance relationships among the phrases in a sentence.

3.2 Quasi-Synchronous Phrase Dependency Translation

Let $\mathcal{X}$ denote the set of all strings in a source language and, for a particular $\boldsymbol{x} \in \mathcal{X}$, let $\mathcal{Y}_{\boldsymbol{x}}$ denote the set of its possible translations (correct and incorrect) in the target language. Given a sentence $\boldsymbol{x}$ and its lexical dependency tree $\tau_{\boldsymbol{x}}$, we formulate the translation problem as finding the target sentence $\boldsymbol{y}^*$, the phrase segmentation π^* of $\boldsymbol{x}$, the phrase segmentation ϕ^* of $\boldsymbol{y}^*$, the phrase dependency tree τ_ϕ^* on the target phrases ϕ^*, and the one-to-one phrase alignment $\boldsymbol{b}^*$ such that

$$\langle \boldsymbol{y}^*, \pi^*, \phi^*, \tau_\phi^*, \boldsymbol{b}^* \rangle = \underset{\langle \boldsymbol{y}, \pi, \phi, \tau_\phi, \boldsymbol{b} \rangle}{\operatorname{argmax}} \boldsymbol{\theta} \cdot \boldsymbol{h}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, \boldsymbol{y}, \pi, \phi, \tau_\phi, \boldsymbol{b}) \quad (1)$$

where $\boldsymbol{h}$ is a vector of feature functions and $\boldsymbol{\theta}$ is a vector of feature weights. The source-language dependency parse $\tau_{\boldsymbol{x}}$ is optional and can be omitted if no source dependency parser is available. If $\tau_{\boldsymbol{x}}$ is provided, we include tree-to-tree configurational features from QG, which are described in Section 5.3. Hence we call the model defined in Equation (1) a **quasi-synchronous phrase dependency** (QPD) translation model.

Our model extends the phrase-based translation model of Koehn, Och, and Marcu (2003). The phrase segmentation variables ϕ and the one-to-one phrase alignment $\boldsymbol{b} : [n'] \rightarrow [n']$ are taken directly from phrase-based translation. For all $i \in [n']$, if $\boldsymbol{b}(i) = j$, then π_j is a subvector of $\boldsymbol{x}$ and ϕ_i is a subvector of $\boldsymbol{y}$. If $\tau_{\boldsymbol{x}}$ is not given and the features ignore τ_ϕ, then the remaining variables ($\boldsymbol{x}$, $\boldsymbol{y}$, π, ϕ, and $\boldsymbol{b}$) are defined in the same way as in phrase-based models.

Computational tractability requires that the feature functions $\boldsymbol{h}$ decompose across "parts" of the output structures in the model. The feature functions that look only at the phrase-based variables ($\boldsymbol{x}$, $\boldsymbol{y}$, π, ϕ, and $\boldsymbol{b}$) are identical to the features used in the Moses phrase-based system (Koehn et al. 2007), so they decompose in the same way as in Moses.[4] For clarity, we partition the features and weights into two parts, namely, $\boldsymbol{\theta} = \langle \boldsymbol{\theta}', \boldsymbol{\theta}'' \rangle$ and $\boldsymbol{h} = \langle \boldsymbol{h}', \boldsymbol{h}'' \rangle$, where $\boldsymbol{\theta}'$ are the weights for the phrase-based features $\boldsymbol{h}'$

3 Further, we restrict our attention to projective phrase dependency trees in this article. We conjecture that non-projective trees may be a better fit for translation modeling (Carreras and Collins 2009; Galley and Manning 2009), particularly for certain language pairs, but we leave their exploration for future work.

4 A more detailed discussion of how the Moses features decompose can be found in Gimpel (2012).

and $\boldsymbol{\theta}''$ are the weights for the QPD features $\boldsymbol{h}''$. So we rewrite the right-hand side of Equation (1) as the following:

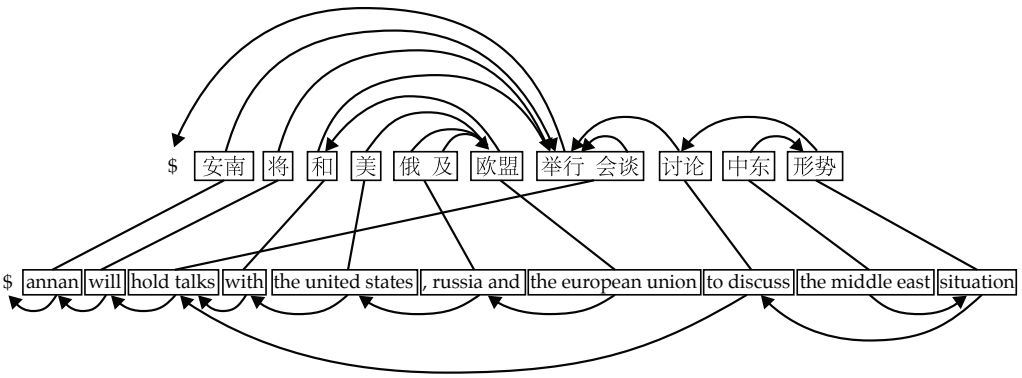
$$\underset{\langle \boldsymbol{y},\pi,\phi,\tau_\phi,\boldsymbol{b}\rangle}{\operatorname{argmax}} \; \boldsymbol{\theta}' \cdot \boldsymbol{h}'(\boldsymbol{x},\boldsymbol{y},\pi,\phi,\boldsymbol{b}) + \boldsymbol{\theta}'' \cdot \boldsymbol{h}''(\boldsymbol{x},\tau_{\boldsymbol{x}},\boldsymbol{y},\pi,\phi,\tau_\phi,\boldsymbol{b}) \tag{2}$$

Furthermore, we assume an additive decomposition across individual phrase dependencies in the phrase dependency tree τ_ϕ, allowing us to rewrite Equation (2) as

$$\begin{aligned}\underset{\langle \boldsymbol{y},\pi,\phi,\tau_\phi,\boldsymbol{b}\rangle}{\operatorname{argmax}} \; & \boldsymbol{\theta}' \cdot \boldsymbol{h}'(\boldsymbol{x},\boldsymbol{y},\pi,\phi,\boldsymbol{b}) \\ & + \sum_{i=1}^{n'} \boldsymbol{\theta}'' \cdot \boldsymbol{f}(\boldsymbol{x},\tau_{\boldsymbol{x}},i,\tau_\phi(i),\phi_i,\phi_{\tau_\phi(i)},\boldsymbol{b}(i),\boldsymbol{b}(\tau_\phi(i)),\pi_{\boldsymbol{b}(i)},\pi_{\boldsymbol{b}(\tau_\phi(i))})\end{aligned} \tag{3}$$

where we introduce new notation $\boldsymbol{f}$ to represent the feature vector that operates on a single phrase dependency at a time in the "arc-factored" decomposition of $\boldsymbol{h}''$. Each feature in $\boldsymbol{f}$ can look at the entirety of $\boldsymbol{x}$ and $\tau_{\boldsymbol{x}}$ because they are inputs, but can only look at a single target-side phrase dependency $\langle \phi_i, \phi_{\tau_\phi(i)}\rangle$ at a time (along with their aligned source phrases $\pi_{\boldsymbol{b}(i)}$ and $\pi_{\boldsymbol{b}(\tau_\phi(i))}$ and the indices).

Example. Figure 5 shows an example. The inputs to the model are a segmented Chinese sentence and its lexical dependency tree. We used the Stanford Chinese word segmenter (Chang, Galley, and Manning 2008) to segment the Chinese data and the Stanford parser (Levy and Manning 2003) to get Chinese dependency trees. The outputs



references: annan to hold talks with us , russia and eu over situation in middle east
annan will discuss middle east situation with u.s. , russia and european union
annan to discuss mideast situation with us , russia and eu
annan to meeting the us , russia and eu to discuss middle east crisis

Figure 5
Example output of our model for Chinese→English translation. The word-segmented Chinese sentence and dependency tree are inputs. Our model's outputs include the English translation, phrase segmentations for each sentence (a box surrounds each phrase), a one-to-one alignment between the English and Chinese phrases, and a projective dependency tree on the English phrases. Note that the Chinese dependency tree is on words whereas the English dependency tree is on phrases.

of the model include a segmentation of the Chinese sentence into phrases, the English translation, its segmentation into phrases, a projective dependency tree on the English phrases, and a one-to-one alignment between the English phrases and Chinese phrases. Four reference translations are also shown. In this example, the model correctly moved the phrase *hold talks* and also noted its connection to *to discuss* by making the latter a phrasal dependent.

4. Rule Extraction

In typical statistical machine translation (SMT) models, the space of allowable translations is constrained by a set of **rules**. Informally, a rule consumes part of the input text and emits text in the output language. Building an SMT system typically requires collecting a massive set of rules from parallel text, a process called **rule extraction**.

For phrase-based translation, these rules are **phrase pairs** and the translation space is constrained by the phrase pairs in the phrase table.[5] In our model, even though we have additional structure (i.e., the phrase dependency tree τ_ϕ), we do not want to enforce any additional constraints on the search space. That is, the space of valid translations is still constrained solely by a standard phrase table. We allow τ_ϕ to be *any* projective phrase dependency tree on ϕ, so the structure of τ_ϕ merely affects how translations are scored, not what translations are permitted. We made this decision because we did not want to reduce the coverage of phrase-based models, which is one of their strengths. Rather, we wanted to better score their translations.[6]

So, even though our phrase dependency rules do not consume parts of the input, we still speak in terms of "rule extraction" because our procedure is similar to rule extraction in other systems and we define feature functions on our rules in a standard way. In particular, we use the extracted rule instances to compute relative frequency estimates for many of the features presented in Section 5.

The rest of this section is organized as follows. In Section 4.1 we describe how we extract rules that only look at target-side words and syntactic structure. In Section 4.2 we extract rules that also look at the source sentence, but not its *syntax*. (Although our system uses unlexicalized *features* based on source-side syntax, they do not derive from rules; we turn to features in Section 5). This lets us avoid the computational expense of parsing the source side of the parallel training corpus.

4.1 Target-Tree Rules

We first extract rules that only consider the target side: $\boldsymbol{y}$, ϕ, and τ_ϕ. These rules can be used as the basis for "dependency language model" features (Shen, Xu, and Weischedel 2008; Galley and Manning 2009; Zhang 2009), though unlike previous work, our features model both the phrase segmentation and dependency structure. Typically, these sorts of features are relative frequencies from a corpus parsed using a supervised parser. However, there do not currently exist treebanks with annotated phrase dependency

5 It is common to add "identity" phrase pairs for unknown words to allow them to pass through untranslated.

6 This strategy can also lead to limitations. Because we do not expand the search space beyond what is licensed by the phrase table, we are limited by the ability of the underlying phrase-based model to provide us with a good search space.

trees. Our solution is to use a standard lexical dependency parser and extract phrase dependencies using bilingual information.[7] Essentially, we combine phrases from the standard phrase extraction pipeline with selected lexical dependencies from the output of a dependency parser.

We first give an overview of our approach and then describe it more formally. We begin by obtaining word alignments and extracting phrase pairs using the standard heuristic approach of Koehn, Och, and Marcu (2003). We then parse the target sentence with a projective dependency parser to obtain a projective dependency tree $\tau_{\boldsymbol{y}}$ for a sentence $\boldsymbol{y}$. Note that $\tau_{\boldsymbol{y}}$ is a tree on *words*, not phrases (cf. τ_ϕ). For each pair of target-side phrases in the phrase pairs from phrase extraction, we extract a phrase dependency (along with its direction) if the phrases do not overlap and there is at least one lexical dependency between them. If there is only a dependency in one direction, we extract a single phrase dependency with that direction. If there are lexical dependencies in both directions, we extract a phrase dependency only for the single longest lexical dependency, and in its direction. Because we use a projective dependency parser, the longest lexical dependency between two phrases is guaranteed to be unique. If a phrase contains a root word in $\tau_{\boldsymbol{y}}$, we extract a phrase dependency with the wall symbol as its head.

We now present the procedure more formally. Given word-aligned sentence pairs, we extract phrase pairs that are **p-consistent** with (i.e., do not violate) the word alignments. Let R denote a relation between the two sets $[n]$ and $[m]$, where $n = |\boldsymbol{x}|$ and $m = |\boldsymbol{y}|$. If a pair (i, j) belongs to R for some $i \in [n]$ and $j \in [m]$, then we say that x_i is aligned to y_j. We define new notation R here instead of using $\boldsymbol{b}$ because R allows many-to-many word alignments, which are typically used for phrase extraction.[8] A phrase pair $\langle \boldsymbol{x}_i^j, \boldsymbol{y}_k^l \rangle$ is **p-consistent** with R if, for all u such that $i \leq u \leq j$, and all v such that (u, v) belongs to R, it is the case that $k \leq v \leq l$. So far this is identical to the phrase extraction pipeline used in Moses.

Given word alignments R and a dependency tree $\tau_{\boldsymbol{y}}$ on $\boldsymbol{y}$, we extract (target-side) **phrase dependencies**. We say a phrase dependency $\langle \boldsymbol{y}_i^j, \boldsymbol{y}_k^l \rangle$ with $\boldsymbol{y}_k^l$ as the parent phrase is **d-consistent** with $\tau_{\boldsymbol{y}}$ and R if:

1. $\exists \boldsymbol{x}_{i'}^{j'}, \boldsymbol{x}_{k'}^{l'}$ such that $\langle \boldsymbol{x}_{i'}^{j'}, \boldsymbol{y}_i^j \rangle$ and $\langle \boldsymbol{x}_{k'}^{l'}, \boldsymbol{y}_k^l \rangle$ are p-consistent with R
2. $\boldsymbol{y}_i^j$ and $\boldsymbol{y}_k^l$ do not overlap: $(1 \leq i \leq j < k \leq l \leq m) \vee (1 \leq k \leq l < i \leq j \leq m)$
3. the longest lexical dependency from $\boldsymbol{y}_i^j$ to $\boldsymbol{y}_k^l$ is longer than the longest from $\boldsymbol{y}_k^l$ to $\boldsymbol{y}_i^j$: $\max_{u: i \leq u \leq j, k \leq \tau_{\boldsymbol{y}}(u) \leq l} |\tau_{\boldsymbol{y}}(u) - u| > \max_{v: k \leq v \leq l, i \leq \tau_{\boldsymbol{y}}(v) \leq j} |\tau_{\boldsymbol{y}}(v) - v|$

The final condition also implies that there is a lexical dependency from a word in $\boldsymbol{y}_i^j$ to a word in $\boldsymbol{y}_k^l$: $\exists u, i \leq u \leq j$, such that $k \leq \tau_{\boldsymbol{y}}(u) \leq l$.

7 Other ways of getting phrase dependencies are possible. For example, for a monolingual task, Wu et al. (2009) used a shallow parser to convert lexical dependencies from a dependency parser into phrase dependencies.

8 Many-to-many word alignments can be obtained from certain alignment models or, more frequently, by using heuristics to combine alignments from one-to-many and many-to-one alignments (Koehn, Och, and Marcu 2003).

We also need to extract root phrase dependencies. We say a root phrase dependency $\langle \boldsymbol{y}_{i'}^{j}, \$ \rangle$ is **d-consistent** with $\tau_{\boldsymbol{y}}$ and R if:

1. $\exists \boldsymbol{x}_{i'}^{j'}$ such that $\langle \boldsymbol{x}_{i'}^{j'}, \boldsymbol{y}_{i}^{j} \rangle$ is p-consistent with R
2. $\exists u, i \leq u \leq j$, such that $\tau_{\boldsymbol{y}}(u) = 0$

We extract all phrase dependencies that are d-consistent with the word alignments and target-side lexical dependency trees. We note that while extracting phrase dependencies we never explicitly commit to any single phrase dependency tree for a target sentence. Rather, we extract phrase dependencies from all phrase dependency trees compatible with the word alignments and the lexical dependency tree. Thus we treat phrase dependency trees analogously to phrase segmentations in phrase extraction.

When actually extracting phrase dependencies, we record additional information from the sentence pairs in which we found them. Specifically, for d-consistent phrase dependencies $\langle \boldsymbol{y}_{i'}^{j}, \boldsymbol{y}_{k}^{l} \rangle$ (where $\boldsymbol{y}_{k}^{l}$ is the parent), we extract tuples of the following form:

$$\langle \boldsymbol{y}_{i'}^{j}, \boldsymbol{y}_{k}^{l}, y_{u^*}, y_{\tau_{\boldsymbol{y}}(u^*)}, \mathbb{I}\left[j < k\right] \rangle \tag{4}$$

where $\mathbb{I}[P]$ is the indicator function that returns 1 if P evaluates to true and 0 otherwise. The index u^* is chosen to make $\langle y_{u^*}, y_{\tau_{\boldsymbol{y}}(u^*)} \rangle$ the longest lexical dependency within the phrase dependency:

$$u^* = \operatorname*{argmax}_{u: i \leq u \leq j, k \leq \tau_{\boldsymbol{y}}(u) \leq l} |\tau_{\boldsymbol{y}}(u) - u| \tag{5}$$

This lexical dependency is recorded for use in back-off features, analogous to the lexical weighting in phrase-based models. The fifth field in Equation (4) holds the direction of the phrase dependency, which is also the direction of the longest lexical dependency. Root phrase dependencies use $k = l = 0$ in the parent phrase and designate \$ as y_0. The direction of root phrase dependencies is inconsequential and can remain as $\mathbb{I}\left[j < k\right]$.

4.1.1 Examples. What do typical phrase dependencies look like? Tables 3 and 4 show some of the most frequent examples of root phrases and parent-child phrase dependencies extracted by this technique on our German–English (DE→EN) corpus. The English side of the parallel corpus was parsed using TurboParser (Martins et al. 2010). Naturally, there are many phrase dependencies with a single word in each phrase, but because these are very similar to lists of frequent lexical dependencies in a parsed corpus, we have only shown dependencies with phrases containing more than one word.

Root phrases (Table 3) frequently contain a subject along with a verb (*it is*, *i would like*, etc.), though the lexical root is typically a verb or auxiliary. These are examples of how we can get syntactic information for phrases that typically would not correspond to constituents in phrase structure trees.

Table 4 shows frequent phrase dependencies from the same corpus; because this corpus is mostly European Parliamentary proceedings, certain formulaic and domain-specific phrases appear with large counts. When phrases attach to each other, they typically behave like their heads. For example, in the phrase dependency *of the*←*union*, the word *union* is the child phrase because *of the* is behaving like *of*. There is likely also a

Table 3
Top 60 most frequent root phrases in DE→EN data with at least two words, shown with their counts. Shown in **bold** are the actual root words in the lexical dependency trees from which these phrases were extracted; these are extracted along with the phrases and used for back-off features.

35,265	it **is**	6,210	i **think**	4,843	**would** be	2,918	**thank** you
13,751	this **is**	6,115	**is** that	4,289	we **will**	2,816	it **will**
12,763	**is** a	6,105	**is** not	4,262	i **believe** that	2,788	**is** to
11,831	we **have**	6,019	, it **is**	4,018	**is** also	2,737	it **is** a
11,551	**would** like	5,975	**believe** that	3,910	that **is** why	2,736	it **has**
11,245	we **must**	5,818	**will** be	3,838	i **would** like to	2,730	they **are**
11,243	**is** the	5,706	we **need**	3,775	**would** like to	2,611	we **can**
11,015	i **would** like	5,628	there **are**	3,505	**hope** that	2,580	i **think** that
10,008	there **is**	5,495	**should** like	3,427	**is** an	2,551	i **will**
8,983	i **am**	5,453	i **should** like	3,239	, i **would** like	2,483	**does** not
8,019	we **are**	5,227	i **hope**	3,130	i **hope** that	2,482	debate **is**
7,722	that **is**	5,150	, **is**	3,101	**need** to	2,445	i **can**
6,883	i **would**	5,110	we **should**	3,059	it **was**	2,438	**want** to
6,443	i **have**	5,010	**has** been	3,021	**have** been	2,416	**must** be
6,328	i **believe**	4,917	**do** not	2,937	**think** that	2,405	this **is** a

dependency from *the* to *union* whenever the longer phrase dependency is extracted, but due to our practice of following the longest lexical dependency in deciding the direction, *of*←*union* is favored over *the*→*union*.

We note that even though these phrase dependencies only contain words from the target language (English), the presence and counts of the phrase dependencies will

Table 4
Most frequent phrase dependencies in DE→EN data, shown with their counts and attachment directions. Child phrases point to their parents. To focus on interesting phrase dependencies, we only show those in which one phrase has at least two tokens and neither phrase is entirely punctuation. The words forming the longest lexical dependency in each extracted phrase dependency are shown in **bold**; these are used for back-off features.

30,064	**mr** → **president** ,	4,582	i **believe** ← **that**
19,931	**the** → european **union**	4,516	, **which** ← **is**
18,819	**the** european → **union**	4,347	**that** ← **will** be
12,318	**i** → **would** like	4,297	the **fact** ← **that**
11,990	**the** → member **states**	4,289	it **is** ← **important**
8,169	it **is** ← **that**	4,232	**one** ← **of** the
7,779	**the** → european **parliament**	4,215	**of** the ← **commission**
7,762	**madam** → **president** ,	3,932	it **is** ← **not**
7,448	**the** european → **parliament**	3,793	**i** → **would** like to
6,897	**of** the ← **union**	3,761	**in** the ← **union**
6,196	**mr** → **president** , i	3,752	**in** ← member **states**
6,188	**i** → **should** like	3,673	**president** → ladies and **gentlemen** ,
6,087	**that** the ← **is**	3,673	**is** ← **that** the
5,478	**i** → **believe** that	3,667	**president** , → ladies and **gentlemen** ,
5,283	**of** the ← european **union**	3,602	i **hope** ← **that**
5,268	**that** → and **that**	3,531	**we** → **need** to
4,956	**of** the european ← **union**	3,495	**the** → **fact** that
4,902	**,** and → **is**	3,494	that **the** → **commission**
4,798	**the** → united **states**	3,462	**i** → **do** not
4,607	**)** mr → **president** ,	3,446	, **the** → **commission**
4,592	**,** it → **is**	3,421	**that** the ← **will**

depend on the source language through the word alignments. For example, when *of the union* is expressed in German, the preposition will often be dropped and the definite article chosen to express genitive case. In our corpus, the most common translation of the English *union* is the German noun *union*, which is feminine. The genitive feminine definite article is *der* and, indeed, we find in the phrase table that the translation of *of the union* with highest probability is *der union*.[9] Thus the dominance of the phrase dependency *of the*←*union* (6,897 occurrences) as compared with *of*←*the union* (142 occurrences) is caused by the German translation.

4.1.2 Word Clusters. When trying to compute feature functions for dependencies between long phrases, we expect to face problems of data sparseness. Long phrases do not occur very often, so *pairs* of long phrases will occur less often still. One way to address this is to also extract rules that use part-of-speech (POS) tags in place of words. However, since words can have multiple POS tags, we would then need to infer POS tags for the words in order to determine which rule is applicable. So we instead use hard word clusters, which provide a deterministic mapping from words to cluster identifiers. Furthermore, certain types of hard word clusters, such as Brown clusters (Brown et al. 1992), have been shown to correspond well to POS tag categories (Christodoulopoulos, Goldwater, and Steedman 2010). We chose Brown clusters for this reason.

Brown clustering uses a bigram hidden Markov model (HMM) in which states are hard cluster labels and observations are words. The emission distributions are constrained such that each word has a nonzero emission probability from at most one cluster label. Clusters can be obtained efficiently through a greedy algorithm that approximately maximizes the HMM's log-likelihood by alternately proposing new clusters and merging existing ones. This procedure actually produces a hierarchical clustering, but we discard the hierarchy information and simply use unique IDs for each cluster. The number of clusters is specified as an input to the algorithm; we used 100 clusters for all experiments in this article. Additional details on cluster generation for our data sets are provided in Appendix B.

Given Brown clusters, we extract tuples like those above in which we replace each word by its Brown cluster ID:

$$\langle \text{clust}(\boldsymbol{y}_i^j), \text{clust}(\boldsymbol{y}_k^l), \text{clust}(y_{u^*}), \text{clust}(y_{\tau_{\boldsymbol{y}}(u^*)}), \mathbb{I}\left[j < k\right] \rangle \quad (6)$$

where clust() is a function that takes a sequence of words and replaces each by its Brown cluster ID. The index u^* is defined as in Equation (5). Examples of frequent Brown cluster phrase dependencies, including root dependencies, are shown in Table 5.

4.2 String-to-Tree Rules

Our simplest probability features use the information in these tuples, but we also extract tuples with more information to support richer features. In particular, we record aligned

9 The phrase table probability of the German *der union* given the English *of the union* is 0.64. The next most-probable German phrase is *der europäischen union*, with probability 0.03.

Table 5
Most frequent Brown cluster phrase dependencies extracted from DE→EN data, shown with their counts. As in Table 4, we only show those in which one phrase has at least two tokens and neither phrase is entirely punctuation. Each cluster is shown as a set of words large enough to cover 95% of the token counts in the cluster, up to a maximum of four words. It is characteristic of Brown clustering that very frequent tokens (e.g., function words) often receive their own clusters.

47,137	{mr, mrs, madam, mr.} → {president, president-in-office, van, barroso} ,
35,656	$ ← it is
29,775	the → {time, way, right, question} of
28,199	the → {european, soviet} {union, parliament, globalisation}
27,373	$ ← i {say, believe, think, know}
26,480	the → {state, development, group, security} of
26,388	the {european, soviet} → {union, parliament, globalisation}
24,726	{one, part, number, behalf} ← of the
22,536	of the ← {people, countries, members, citizens}
21,449	{state, development, group, security} {and, or} → {state, development, group, security}
21,007	$ ← {we, they} {should, must, cannot, shall}
20,933	{state, development, group, security} → {and, or} {state, development, group, security}
20,919	the → {one, part, number, behalf} of
20,897	of the ← {report, committee, issue, agreement}
20,081	the → {economic, political, international, national} {policy, years, rights, market}
19,209	the → {report, committee, issue, agreement} of
18,535	{people, countries, members, citizens} ← of the
18,523	$ ← {say, believe, think, know} that
18,510	{time, way, right, question} ← of the
18,232	the → {member, united} {states, nations}
18,157	{one, part, number, behalf} of ← {people, countries, members, citizens}
17,950	{people, countries, members, citizens} {and, or} → {people, countries, members, citizens}
17,643	{state, development, group, security} ← of the
17,539	the → {people, countries, members, citizens} of
17,457	the → {economic, political, international, national} {state, development, group, security}
16,608	to {take, make, see, help} ← {people, countries, members, citizens}
16,163	the {time, way, right, question} ← of
15,517	the → {economic, political, international, national} {people, countries, members, citizens}
15,292	in the ← {report, committee, issue, agreement}
15,257	a → {new, good, u.s., common} {report, committee, issue, agreement}
15,223	the {state, development, group, security} ← of
15,217	{people, countries, members, citizens} → {and, or} {people, countries, members, citizens}
15,214	it is ← {important, clear, necessary, concerned}
14,977	i → {say, believe, think, know} that
14,697	$ ← is {important, clear, necessary, concerned}
14,582	i {say, believe, think, know} ← that
14,399	{should, must, cannot, shall} ← be {made, taken, put, set}
14,146	$ ← this is
14,089	a {new, good, u.s., common} → {report, committee, issue, agreement}
14,047	{europe, china, today, women} → {and, or} {europe, china, today, women}
13,599	{made, taken, put, set} ← {by, from, into, between} the
13,190	the → {new, good, u.s., common} {report, committee, issue, agreement}
13,089	the → {new, good, u.s., common} {people, countries, members, citizens}
13,035	$ ← {we, they} have
13,034	{economic, political, international, national} → {and, or} {economic, political, international,. . . }
13,013	$ ← is a
12,713	$ ← {need, want, needs, wish} to
12,399	$ ← i {say, believe, think, know} that
12,387	the → {time, way, right, question} of the
12,319	i → would {like, according, relating}
12,217	in the ← {eu, world, government, country}
12,125	the → {economic, political, international, national} {report, committee, issue, agreement}
11,979	of ← {economic, political, international, national} {state, development, group, security}
11,955	the {report, committee, issue, agreement} ← of
11,838	$ ← {we, they} {are, were}
11,551	$ ← would {like, according, relating}
11,537	the → {people, countries, members, citizens} of the

source phrases and details about reordering and the presence of gaps between phrases. That is, for d-consistent phrase dependencies $\langle \boldsymbol{y}_{i'}^{j}, \boldsymbol{y}_{k}^{l} \rangle$, we extract tuples

$$\begin{aligned}
&\langle \boldsymbol{y}_{i'}^{j}, \boldsymbol{y}_{k}^{l}, \boldsymbol{x}_{i''}^{j'}, \boldsymbol{x}_{k''}^{l'}, \\
&\mathbb{I}\left[j < k\right], \\
&\mathbb{I}\left[\mathbb{I}\left[j < k\right)\right] = \mathbb{I}\left[j' < k'\right]\right], \\
&\mathbb{I}\left[(j+1 = k) \vee (l+1 = i)\right], \\
&\mathbb{I}\left[(j'+1 = k') \vee (l'+1 = i')\right]\rangle
\end{aligned} \tag{7}$$

for all i', j', k', and l' such that the phrase pairs $\langle \boldsymbol{x}_{i''}^{j'}, \boldsymbol{y}_{i}^{j} \rangle$ and $\langle \boldsymbol{x}_{k''}^{l'}, \boldsymbol{y}_{k}^{l} \rangle$ are p-consistent with R, and such that $\boldsymbol{x}_{i''}^{j'}$ does not overlap with $\boldsymbol{x}_{k''}^{l'}$.[10] Again, $\mathbb{I}[P]$ is the indicator function that returns 1 if P evaluates to true and 0 otherwise. That is, we include the two target phrases, their aligned source phrases, the direction of the target attachment, the **orientation** between the source and target phrases (whether the two target phrases are in the same order as their aligned source phrases or swapped), whether a gap is present between the two target phrases, and finally whether a gap is present between the two source phrases. When $\boldsymbol{y}_{k}^{l} = \$$, all of the additional fields are irrelevant except the aligned source phrase $\boldsymbol{x}_{i''}^{j'}$.

We now note some examples of the phenomena that we can model with these richer tuples. A common cause of reordering in German-to-English translation relates to verbs. Figure 6 shows two examples of frequently extracted phrase dependencies that model verb movement. Figure 6(a) gives an example of how German reorders the finite verb to the end of a dependent clause, whereas English keeps it next to the subject. The extracted rule, shown below the sentence pair, only applies when intervening words appear on the German side and no intervening words appear on the English side. This is indicated by the presence (absence) of an ellipsis on the German (English) side of the rule.

Figure 6(b) shows an example of how German moves an infinitive (*danken*, "to thank") to the end of an independent clause when a modal verb (*möchte*, "would like") is present. The ellipses on both sides indicate that other words must be present between both the source and target phrase pairs. We note that this rule says nothing about what fills the gap. In particular, the gap-filling material does *not* have to be translationally equivalent, and indeed in the given sentence pair it is not. As opposed to rules in hierarchical phrase-based models (Chiang 2005), which typically specify translationally equivalent substructures, this rule simply models the reordering and long-distance movement of the infinitive. Much prior work has found phrase pairs with gaps to be useful for machine translation (Simard et al. 2005; Crego and Yvon 2009; Galley and Manning 2010), and we extract tuples as in Equation (7) so that we can model such structures, even though we do not directly model gap-filling like hierarchical models and other models based on synchronous context-free grammar (Zollmann and Venugopal 2006, inter alia).

10 This non-overlapping constraint is what differentiates these tuples from the target-tree rule tuples from the previous section, which are extracted even when the source phrases overlap.

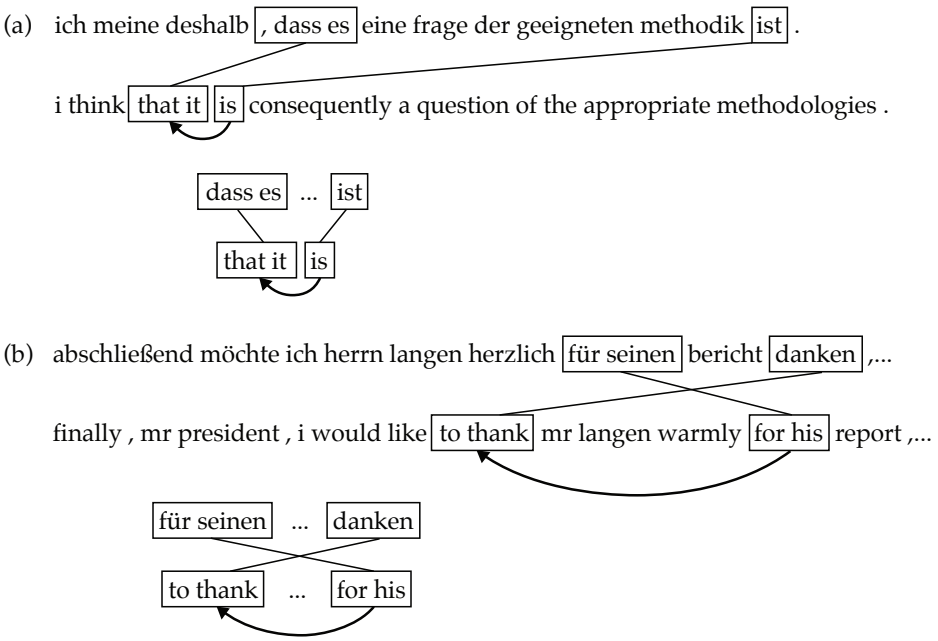


Figure 6
Examples of illustrative sentence pairs and frequently extracted rules that model verb movement between German and English. An ellipsis indicates that there must be material between the two phrases for the rule to apply. (a) Example of movement of the finite verb to the end of a dependent clause. (b) Example of movement of an infinitive to the end of an independent clause following a modal verb (*möchte*, 'would like'). Discussion of the features used to score these string-to-tree rules is given in Section 5.2.

The tuples described here are used to compute all of the lexicalized phrase dependency features in our model. We extract each tuple with a count of 1 each time it is observed, aggregate the counts across all sentence pairs in the parallel corpus, and use the counts to compute the statistical features we present in the next section. We also have structural features that consider string-to-tree and tree-to-tree configurations, but these do not require any rule extraction. In the next section we describe the full set of features in our model.

5. Features

Our model extends the phrase-based translation model of Moses (Koehn et al. 2007), so we include all of its features in our model. These include four phrase table probability features, a phrase penalty feature, an n-gram language model, a distortion cost, six lexicalized reordering features, and a word penalty feature. These features are contained in $\boldsymbol{h}'$ in Equation (3), reproduced here:

$$\underset{\langle \boldsymbol{y}, \pi, \phi, \tau_\phi, \boldsymbol{b} \rangle}{\operatorname{argmax}} \quad \boldsymbol{\theta}' \cdot \boldsymbol{h}'(\boldsymbol{x}, \boldsymbol{y}, \pi, \phi, \boldsymbol{b}) + \sum_{i=1}^{n'} \boldsymbol{\theta}'' \cdot \boldsymbol{f}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, i, \tau_\phi(i), \phi_i, \phi_{\tau_\phi(i)}, \boldsymbol{b}(i), \boldsymbol{b}(\tau_\phi(i)), \pi_{\boldsymbol{b}(i)}, \pi_{\boldsymbol{b}(\tau_\phi(i))}) \tag{8}$$

We now describe in detail the additional features $\boldsymbol{f}$ that are used to score phrase dependency trees. Each operates on a single phrase dependency and takes the arguments $\langle \boldsymbol{x}, \tau_{\boldsymbol{x}}, c, d, \phi_c, \phi_d, c', d', \pi_{c'}, \pi_{d'} \rangle$, which are, in order, the source sentence ($\boldsymbol{x}$), the source dependency tree ($\tau_{\boldsymbol{x}}$), the target child phrase index (c), the target parent phrase index (d), the target child phrase (ϕ_c), the target parent phrase (ϕ_d), the index of the source phrase aligned to the target child (c'), the index of the source phrase aligned to the target parent (d'), the child-aligned source phrase ($\pi_{c'}$), and the parent-aligned source phrase ($\pi_{d'}$).

Like the phrase probability features in Moses, many of our feature functions are conditional probabilities computed using relative frequency estimation given the full collection of extracted tuples. That is, for a tuple $\langle \alpha, \beta \rangle$, the conditional probability of field α given field β is estimated as

$$\tilde{p}(\alpha \mid \beta) = \frac{\#\{\langle \alpha, \beta \rangle\}}{\sum_{\beta'} \#\{\langle \alpha, \beta' \rangle\}} \tag{9}$$

where $\#\{\langle \alpha, \beta \rangle\}$ denotes the count of the tuple $\langle \alpha, \beta \rangle$ in the multiset of extracted tuples. We use the notation $\tilde{p}$ in the following to indicate that relative frequency estimates are being used.[11]

5.1 Target-Tree Features

We first include features that only consider the target-side words and phrase dependency tree; these are computed based on the rules extracted in Section 4.1. The first feature is the sum of the scaled log-probabilities of each phrase dependency attachment in τ_ϕ:

$$f_{\text{pdep}}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, c, d, \phi_c, \phi_d, c', d', \pi_{c'}, \pi_{d'}) = \max\left(0, C + \log \tilde{p}(\phi_c \mid \phi_d, \text{dir}(c, d))\right) \tag{10}$$

where $\text{dir}(c, d)$ is defined

$$\text{dir}(c, d) = \begin{cases} \textsf{root} & \text{if } d = 0 \\ \textsf{left} & \text{if } d > c \\ \textsf{right} & \text{otherwise} \end{cases} \tag{11}$$

and returns the direction of the attachment for head index d and child index c, that is, the direction in which the child resides; root indicates that phrase c is the root.

Although we use log-probabilities in this feature function, we add a constant C, chosen to ensure the feature value is never negative. The reasoning here is that whenever we use a phrase dependency that we have observed in the training data, we want to boost the score of the translation. If we used log-probabilities, each observed dependency would incur a penalty. The max expression prevents unseen parent-child phrase dependencies from causing the score to be negative infinity. Our motivation is a desire for the features to prefer one derivation over another but not to rule out a derivation completely if it merely happens to contain an unseen phrase dependency.

11 Note that, as is standard across many SMT models, all frequencies here are counts of *extraction* events. They are not counts of *derivation* or *translation* events, since many competing rules may be extracted from each training instance.

Because we will use this same practice for all other probability features, we introduce some shorthand for simplicity of presentation. We first redefine this feature:

$$f_{\text{pdep}}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, c, d, \phi_c, \phi_d, c', d', \pi_{c'}, \pi_{d'}) = \max\left(0, C_{\text{pdep}} + \log g_{\text{pdep}}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, c, d, \phi_c, \phi_d, c', d', \pi_{c'}, \pi_{d'})\right) \quad (12)$$

where

$$g_{\text{pdep}}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, c, d, \phi_c, \phi_d, c', d', \pi_{c'}, \pi_{d'}) = \tilde{p}(\phi_c \mid \phi_d, \text{dir}(c, d)) \quad (13)$$

In what follows, we will restrict our attention to defining the g-style functions for probability features, and assume that there is always a corresponding f that has the same subscript and takes the same inputs, as in Equation (12). Furthermore, when presenting the remaining features, we will suppress the arguments of each for clarity; all take the same arguments as f_{pdep} and g_{pdep}.

We will assume C is chosen appropriately for each g based on the minimum log-probability for the feature. For example,

$$C_{\text{pdep}} = 0.01 - \min_{\phi, \phi', r} \log \tilde{p}(\phi \mid \phi', r) \quad (14)$$

that is, the minimum log-probability is found, negated, and a small positive value (0.01) is added to ensure the feature is greater than zero. This ensures that, if a phrase dependency has been seen, its contribution is at least 0.01.

To counteract data sparseness, we include other features that are less specific than g_{pdep}. First, we include a version of this feature with words replaced by Brown clusters:

$$g_{\substack{\text{pdep} \\ \text{clust}}} = \tilde{p}(\text{clust}(\phi_c) \mid \text{clust}(\phi_d), \text{dir}(c, d)) \quad (15)$$

We also include lexical weighting features similar to those used in phrase-based machine translation (Koehn, Och, and Marcu 2003). These use the longest lexical dependencies extracted during rule extraction. First, for all ⟨child, parent, direction⟩ *lexical* dependency tuples $\langle y, y', r \rangle$ in the parsed target side of the parallel corpus, we estimate conditional probabilities $\tilde{p}_{\text{lex}}(y \mid y', r)$ using relative frequency estimation.

Then, assuming the given phrase dependency $\langle \phi_c, \phi_d \rangle$ has longest child-parent lexical dependency $\langle y, y' \rangle$ for direction $\text{dir}(c, d)$, we include the feature:

$$g_{\text{ldep}} = \tilde{p}_{\text{lex}}(y \mid y', \text{dir}(c, d)) \quad (16)$$

We include an analogous feature with words replaced by Brown clusters. Different instances of a phrase dependency may have different lexical dependencies extracted with them. We only use the lexical weight for the most frequent, breaking ties by choosing the lexical dependency that maximizes $\tilde{p}_{\text{lex}}(y \mid y', r)$, as was done similarly by Koehn, Och, and Marcu (2003).

So far we described four features that consider $\boldsymbol{y}$, ϕ, and τ_ϕ: one for phrase dependencies, one for lexical dependencies, and the same two features computed on a transformed version of the corpus in which each word is replaced by its Brown cluster ID.

5.2 String-to-Tree Features

We next discuss features that consider properties of the source sentence $\boldsymbol{x}$, its phrase segmentation π, and the phrase alignment $\boldsymbol{b}$, in addition to $\boldsymbol{y}$, ϕ, and τ_ϕ. However, these features still do not depend on the source tree $\tau_{\boldsymbol{x}}$, so they can be included even when a parser for the source language is not available. We will discuss features that use $\tau_{\boldsymbol{x}}$ in Section 5.3.

These features are similar to the previously defined g_{pdep}, but condition on additional pieces of structure. All features condition on direction. The first pair of features condition on the source phrase ($\pi_{c'}$) aligned to the child phrase (ϕ_c) in the target phrase dependency ($\langle \phi_c, \phi_d \rangle$):

$$g_{\substack{\text{pdep} \\ \text{child}}} = \tilde{p}(\phi_c \mid \phi_d, \text{dir}(c,d), \pi_{c'}) \tag{17}$$

$$g_{\substack{\text{pdep} \\ \text{child} \\ \text{clust}}} = \tilde{p}(\phi_c \mid \text{clust}(\phi_d), \text{dir}(c,d), \pi_{c'}) \tag{18}$$

In the second feature, we condition on word clusters for the parent phrase ϕ_d, but on words in the aligned source phrase $\pi_{c'}$. Because Brown clusters often correspond to syntactic clusters, even at times resembling part-of-speech tags (Christodoulopoulos, Goldwater, and Steedman 2010), it did not seem logical to model translation probabilities between source- and target-language word clusters. This is why we did not include a feature like the above with word clusters for ϕ_c and $\pi_{c'}$. Our use of these clusters is a simple kind of backoff or smoothing that allows some sharing across specific phrases, since statistics on phrase pairs are expected to be sparse.

The next set of features includes those that condition on the orientation between the source- and target-side phrases. The ori function returns the orientation of the aligned source phrases in a target phrase dependency attachment, namely, whether the aligned source phrases are in the same order as the target phrases ("same") or if they are in the opposite order ("swap"):

$$\text{ori}(c,d,c',d') = \begin{cases} \textsf{root} & \text{if } d = 0 \\ \textsf{same} & \text{if } \text{dir}(c,d) = \text{dir}(c',d') \\ \textsf{swap} & \text{otherwise} \end{cases} \tag{19}$$

Given this definition of ori, we define the following features that condition on orientation (in addition to other fields):

$$g_{\substack{\text{pdep} \\ \text{orient}}} = \tilde{p}(\phi_c \mid \phi_d, \text{dir}(c,d), \text{ori}(c,d,c',d')) \tag{20}$$

$$g_{\substack{\text{pdep} \\ \text{orient} \\ \text{clust}}} = \tilde{p}(\text{clust}(\phi_c) \mid \text{clust}(\phi_d), \text{dir}(c,d), \text{ori}(c,d,c',d')) \tag{21}$$

$$g_{\substack{\text{pdep} \\ \text{child} \\ \text{orient}}} = \tilde{p}(\phi_c \mid \phi_d, \text{dir}(c,d), \pi_{c'}, \text{ori}(c,d,c',d')) \tag{22}$$

$$g_{\substack{\text{pdep} \\ \text{child} \\ \text{orient} \\ \text{clust}}} = \tilde{p}(\phi_c \mid \text{clust}(\phi_d), \text{dir}(c,d), \pi_{c'}, \text{ori}(c,d,c',d')) \tag{23}$$

where the last two features condition on the aligned child phrase $\pi_{c'}$ in addition to the direction and orientation.

We next give features that condition on the presence of gaps between the child and parent target phrases and gaps between the aligned phrases on the source side. The $\text{gap}(c, d)$ function indicates whether there is a gap between the phrases indexed by c and d:

$$\text{gap}(c, d) = \begin{cases} \text{root} & \text{if } d = 0 \\ \text{yes} & \text{if } |d - c| \geq 1 \\ \text{no} & \text{otherwise} \end{cases} \tag{24}$$

Given this gap function, we define the following features:

$$g^{\text{pdep}}_{\substack{\text{orient} \\ \text{gap}}} = \tilde{p}(\phi_c \mid \phi_d, \text{dir}(c, d), \text{ori}(c, d, c', d'), \text{gap}(c, d), \text{gap}(c', d')) \tag{25}$$

$$g^{\text{pdep}}_{\substack{\text{orient} \\ \text{gap} \\ \text{clust}}} = \tilde{p}(\text{clust}(\phi_c) \mid \text{clust}(\phi_d), \text{dir}(c, d), \text{ori}(c, d, c', d'), \text{gap}(c, d), \text{gap}(c', d')) \tag{26}$$

All the features mentioned so far have the child phrase on the left-hand side of the conditioning bar. We now present features that have both the child and parent phrases on the left-hand side:

$$g^{\text{pdep}}_{\text{pc}} = \tilde{p}(\phi_c, \phi_d, \text{dir}(c, d) \mid \pi_{c'}, \pi_{d'}) \tag{27}$$

$$g^{\text{pdep}}_{\substack{\text{pc} \\ \text{orient}}} = \tilde{p}(\phi_c, \phi_d, \text{dir}(c, d) \mid \pi_{c'}, \pi_{d'}, \text{ori}(c, d, c', d')) \tag{28}$$

$$g^{\text{pdep}}_{\substack{\text{pc} \\ \text{orient} \\ \text{gap}}} = \tilde{p}(\phi_c, \phi_d, \text{dir}(c, d), \text{gap}(c, d) \mid \pi_{c'}, \pi_{d'}, \text{ori}(c, d, c', d'), \text{gap}(c', d')) \tag{29}$$

These last features score larger rules composed of two phrase pairs from the phrase table. Including direction, orientation, and gaps enables us to model longer-distance reorderings; we showed some examples of such frequently extracted phrase dependencies in Section 4.2.

In all, we introduced 11 features in this section, giving us a total of 15 so far. For the feature ablation experiments in Section 7, we will partition these features into two parts: We refer to the six features with subscript clust as CLUST and the other nine as WORD.

5.2.1 String-to-Tree Configurations (CFG). We now present features that count instances of local reordering configurations involving phrase dependencies. We refer to the features described in this section and the next section as CFG. These features consider the target segmentation ϕ, the target phrase dependency tree τ_ϕ, and the phrase alignment $\boldsymbol{b}$, but not the target words $\boldsymbol{y}$ or the source words $\boldsymbol{x}$, segmentation π, or dependency tree $\tau_{\boldsymbol{x}}$.

Our first set of features only looks at configurations involving direction and orientation. The first feature value is incremented if the child is to the left and the aligned source-side phrases are in the same order:

$$f_{\substack{\text{left}\\\text{same}}} = \mathbb{I}\left[\text{dir}(c,d) = \textsf{left} \wedge \text{ori}(c,d,c',d') = \textsf{same}\right] \tag{30}$$

Another feature fires if the aligned source phrases are in the opposite order:

$$f_{\substack{\text{left}\\\text{swap}}} = \mathbb{I}\left[\text{dir}(c,d) = \textsf{left} \wedge \text{ori}(c,d,c',d') = \textsf{swap}\right] \tag{31}$$

Analogous features are used when the child is to the right of the parent:

$$f_{\substack{\text{right}\\\text{same}}} = \mathbb{I}\left[\text{dir}(c,d) = \textsf{right} \wedge \text{ori}(c,d,c',d') = \textsf{same}\right] \tag{32}$$

$$f_{\substack{\text{right}\\\text{swap}}} = \mathbb{I}\left[\text{dir}(c,d) = \textsf{right} \wedge \text{ori}(c,d,c',d') = \textsf{swap}\right] \tag{33}$$

These four configuration features are shown in order in the leftmost column in Figure 7. They are agnostic as to the presence of gaps between the two target phrases and between the two source phrases. We include 16 features that add gap information to these four coarse configurations, as shown in the remainder of the table. Four gap configurations are possible, constructed from one binary variable indicating the presence or absence of a source gap paired with a binary variable indicating the presence or absence of a target gap. We replicate the four coarse features for each gap configuration, giving us a total of 20 string-to-tree configuration features, all shown in Figure 7.

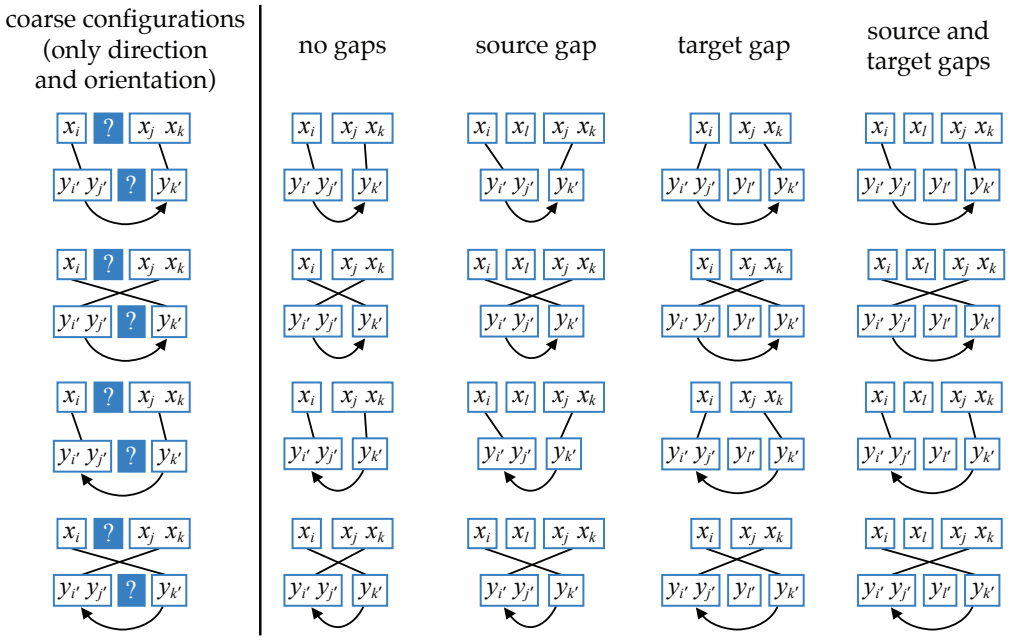


Figure 7
String-to-tree configurations; each is associated with a feature that counts its occurrences in a derivation.

5.2.2 Dependency Length Features. Related to the string-to-tree configurations are features that score source- and target-side lengths (i.e., number of words crossed) of target-side phrase dependencies. These lengths can also be useful for hard constraints to speed up inference; we return to this in Section 6. These features and constraints are similar to those used in vine grammar (Eisner and Smith 2005).

We first include a feature that counts the number of source-side words between the aligned source phrases in each attachment in τ_ϕ. Letting $\pi_{c'} = \boldsymbol{x}_{i'}^{j'}$ and $\pi_{d'} = \boldsymbol{x}_{k'}^{l'}$:

$$f_{\substack{\text{src}\\\text{vine}}} = \mathbb{I}\left[\text{dir}(c', d') = \textsf{left}\right]\left(k' - (j' + 1)\right) + \mathbb{I}\left[\text{dir}(c', d') = \textsf{right}\right]\left(i' - (l' + 1)\right) \quad (34)$$

Although this feature requires the segmentation of the source sentence in order to determine the number of source words crossed, the actual identities of those words are not needed, so the feature does not depend on $\boldsymbol{x}$. We would expect this feature's weight to be negative for most language pairs, encouraging closeness in the source sentence of phrases aligned to each phrase dependency in the target.

We would like to use a similar feature for target-side dependency lengths, for example, where $\phi_c = \boldsymbol{y}_i^j$ and $\phi_d = \boldsymbol{x}_k^l$:

$$\mathbb{I}\left[\text{dir}(c, d) = \textsf{left}\right]\left(k - (j + 1)\right) + \mathbb{I}\left[\text{dir}(c, d) = \textsf{right}\right](i - (l + 1)) \quad (35)$$

However, such a feature could require looking at the entire phrase segmentation being generated to score a single phrase dependency (e.g., if $\tau_\phi(1) = n'$). Using this feature would prevent us from being able to use dynamic programming for decoding (we discuss our approach to decoding in Section 6). Instead, we use a feature that considers *bounds* on the number of target words crossed by each phrase dependency. In particular, the feature sums the maximum number of target words that could be crossed by a particular phrase dependency. We will discuss how this feature is computed when we discuss decoding in Section 6.

We use CFG to refer to the set containing the 20 string-to-tree configuration features and the 2 string-to-tree dependency length features. Adding these 22 features to the 15 from Sections 5.1 and 5.2 gives us 37 QPD features so far.

5.3 Tree-to-Tree Features (TREETOTREE)

The last two sets of features consider the source-side dependency tree $\tau_{\boldsymbol{x}}$ in addition to $\boldsymbol{x}$, π, $\boldsymbol{b}$, $\boldsymbol{y}$, ϕ, and τ_ϕ. These are the only features that use source and target syntax simultaneously. We use TREETOTREE to refer to these features.

5.3.1 Quasi-Synchronous Tree-to-Tree Configurations. We begin with features based on the quasi-synchronous configurations from Smith and Eisner (2006), shown for lexical dependency trees in Figure 8. For a child-parent dependency on the target side, these configurations consider the relationship between the aligned source words. For example, if the aligned source words form a child-parent dependency in the source tree, then we have a "parent-child" configuration. There is also an "other" category for those that do not fit any of the named categories.

However, for our model we need to score configurations involving phrase dependencies. That is, for a child-parent phrase dependency $\langle \phi_c, \phi_d \rangle$ in τ_ϕ, we consider the relationship between $\pi_{c'}$ and $\pi_{d'}$, the source-side phrases to which ϕ_c and ϕ_d align.

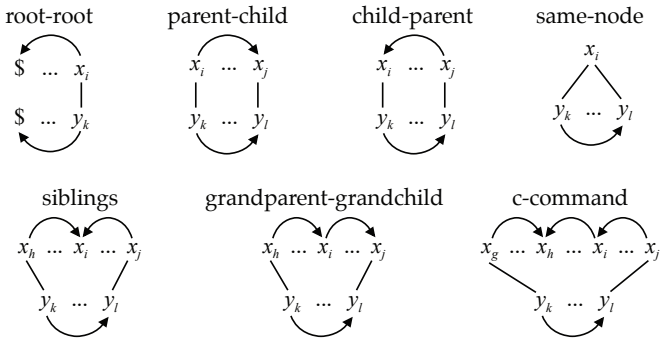


Figure 8
Quasi-synchronous tree-to-tree configurations from Smith and Eisner (2006). There are additional configurations involving NULL alignments and an "other" category for those that do not fit into any of the named categories.

There are several options for computing configuration features for our model, since we use a phrase dependency tree for the target sentence, a lexical dependency tree for the source sentence, and a phrase alignment.

We use a heuristic approach. First we find the full set of configurations that are present between any word in one source phrase and any word in the other source phrase. That is, given a pair of source words, one with index j in source phrase d' and the other with index k in source phrase c', we have a parent-child configuration if $\tau_{\boldsymbol{x}}(k) = j$; if $\tau_{\boldsymbol{x}}(j) = k$, a child-parent configuration is present. In order for the grandparent-grandchild configuration to be present, the intervening parent word must be outside both phrases. For sibling configurations, the shared parent must also be outside both phrases. In lieu of standard (non-sibling) c-command relationships, we define a modified c-command category as follows. We first find the highest ancestors of words j and k that are still in their respective phrases. Of these two ancestors, if neither is an ancestor of the other and if they are not siblings, then the "c-command" feature fires.

After obtaining a list of all configurations present for each pair of words $\langle j, k \rangle$, we fire the feature for the single configuration corresponding to the maximum distance $|j - k|$. If no configurations are present between any pair of words, the "other" feature fires. Therefore, only one configuration feature fires for each extracted phrase dependency attachment.

For the six configurations other than "root-root," we actually include multiple instances of each configuration feature: one set includes direction ($6 \times 2 = 12$ features), another set includes orientation (12 features), and the final set includes both source- and target-side gap information (24 features). There are therefore 49 features in this category (including the single "root-root" feature).

5.3.2 Tree-to-Tree Dependency Path Length Features. Finally, we include features that consider the dependency path length between the source phrases aligned to the target phrases in each phrase dependency. The features in Section 5.2.2 considered distance along the source *sentence* (the number of words crossed). Now we add features that consider distance along the source *tree* (the number of lexical dependency arcs crossed). We expect the learned weights for these features to encourage short dependency path lengths on the source side.

We first include a feature that sums, for each target phrase i, the inverse of the minimum undirected path length between each word in $\pi_{c'} = \boldsymbol{x}_{i'}^{j'}$ and each word in $\pi_{d'} = \boldsymbol{x}_{k'}^{l'}$:

$$f_{\substack{\text{undir}\\\text{path}}} = \sum_{j=i'}^{j'} \sum_{k=k'}^{l'} \frac{1}{\text{minUndirPathLen}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, j, k)} \tag{36}$$

where minUndirPathLen$(\boldsymbol{x}, \tau_{\boldsymbol{x}}, j, k)$ returns the shortest undirected dependency path length from x_j to x_k in $\tau_{\boldsymbol{x}}$. The shortest undirected path length is defined as the number of dependency arcs that must be crossed to travel from one word to the other along the arcs in $\tau_{\boldsymbol{x}}$.

Assuming an analogous function minDirPathLen$(\boldsymbol{x}, \tau_{\boldsymbol{x}}, j, k)$ that computes the minimum *directed* dependency path length, we also include the following feature:

$$f_{\substack{\text{dir}\\\text{path}}} = \sum_{j=i'}^{j'} \sum_{k=k'}^{l'} \frac{1}{\text{minDirPathLen}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, j, k)} \tag{37}$$

If there is no directed path from x_j to x_k, minDirPathLen returns ∞.

Adding these two features gives us a total of 88 QPD features. Along with the 14 phrase-based features there are a total of 102 features in our model.

6. Decoding

For our model, decoding consists of solving Equation (1)—that is, finding the highest-scoring tuple $\langle \boldsymbol{y}, \pi, \phi, \tau_\phi, \boldsymbol{b} \rangle$ for an input sentence $\boldsymbol{x}$ and its parse $\tau_{\boldsymbol{x}}$. This is a challenging search problem, because it is at least as hard as the search problem for phrase-based models, which is intractable (Koehn, Och, and Marcu 2003). Because of this we use a coarse-to-fine strategy for decoding (Charniak and Johnson 2005; Petrov 2009). Coarse-to-fine inference is a general term for procedures that make two (or more) passes over the search space, pruning the space with each pass. Typically, feature complexity is increased in each pass, as richer features can often be computed more easily in the smaller search space.

One simple coarse-to-fine procedure for our model would start by generating a k-best list of derivations using a phrase-based decoder. This "coarse model" would account for all of the phrase-based features. Then we could parse each derivation to incorporate the QPD features and rerank the k-best list with the modified scores; this is the "fine model." The advantage of this approach is its simplicity, but other research has shown that k-best lists for structured prediction tend to have very little diversity (Huang 2008), and we expect even less diversity in cases like machine translation where latent variables are almost always present. Instead, we generate a **phrase lattice** (Ueffing, Och, and Ney 2002) in a coarse pass and perform **lattice dependency parsing** as the fine pass.

The remainder of this section is laid out as follows. We begin by reviewing phrase lattices in Section 6.1. In Section 6.2 we present our basic lattice dependency parsing algorithm. We give three ways to speed it up in Section 6.3; one enables a more judicious search without affecting the search space, and the other two prune the search space in different ways. In Section 6.4, we discuss how decoding affects learning of the feature weights $\boldsymbol{\theta}$, and we describe the structured support vector machine reranking formulation from Yadollahpour, Batra, and Shakhnarovich (2013) that we use. We close

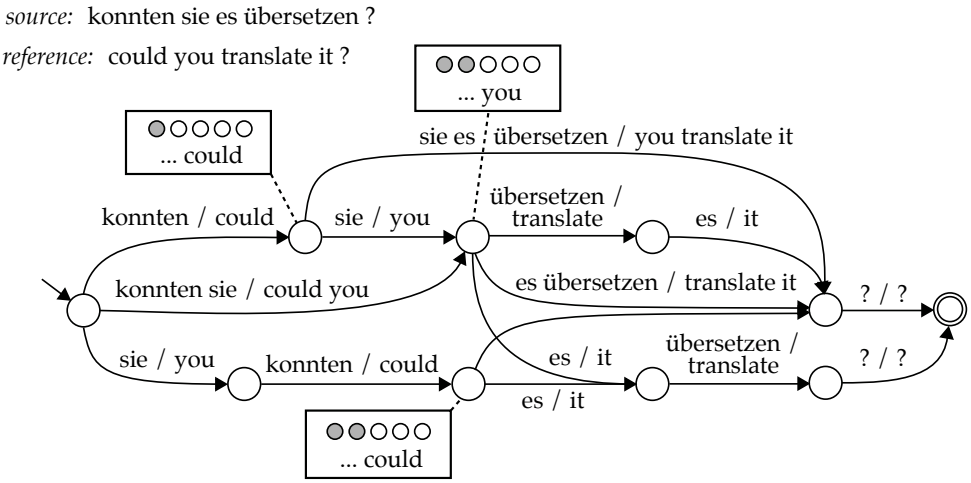


Figure 9
Example phrase lattice for the source sentence shown. Each node contains an n-gram history for computing n-gram language model features and a coverage vector representing the source words that have been translated so far. For clarity, the n-gram history ($n = 2$) and coverage vector are only shown for three nodes.

in Section 6.5 with a brief discussion of how this decoder differs from earlier versions published in Gimpel and Smith (2009b, 2011).

6.1 Phrase Lattices

The most common decoding strategy for phrase-based models is to use beam search (Koehn, Och, and Marcu 2003). The search is performed by choosing phrase pairs from the phrase table and applying them to translate source phrases into the target language. **Coverage vectors** are maintained during decoding to track which words have been translated so far. They are used to enforce the constraint that each source word appear in exactly one phrase pair.

It is often convenient to build a packed representation of the (pruned) search space explored during decoding. For phrase-based models, this representation takes the form of a **phrase lattice** (Ueffing, Och, and Ney 2002), a finite-state acceptor in which each path corresponds to a derivation. Figure 9 shows an example. The source sentence and a reference translation are shown at the top of the figure. Each path from the start node on the left to a final node corresponds to a complete output in the model's output space. Each lattice edge corresponds to a phrase pair used in the output. All paths leading to a given node in the lattice must agree in the set of source words that have been translated thus far. So, every node in the lattice is annotated with the coverage vector of all paths that end there. This is shown for three of the nodes in the figure.

The lattice is constructed such that all features in the model are locally computable on individual lattice edges. To make n-gram language model features local, all paths leading to a given node must end in the same $n - 1$ words.[12] In the example, there are two nodes with equivalent coverage vectors that are separated because they end in

12 In practice, this state replication can be reduced by exploiting sparsity in the language model (Li and Khudanpur 2008).

different words (*you* vs. *could*). Decoders like Moses can output phrase lattices like these; the lattice simply encodes the paths explored during the beam search.

6.2 Lattice Dependency Parsing

Each path in a phrase lattice corresponds to a tuple $\langle \boldsymbol{y}, \pi, \phi, \boldsymbol{b} \rangle$ for the input $\boldsymbol{x}$. To also maximize over τ_ϕ, we perform **lattice dependency parsing**, which allows us to search over the space of tuples $\langle \boldsymbol{y}, \pi, \phi, \boldsymbol{b}, \tau_\phi \rangle$. Lattice parsing jointly maximizes over paths through a lattice and parse structures on those paths.

Because we use an arc-factored phrase dependency model (Equation (3)), the lattice dependency parsing algorithm we use is a straightforward generalization of the arc-factored dynamic programming algorithm from Eisner (1996). The algorithm is shown in Figure 10. It is shown as a set of recursive equations in which shapes are used in place of function names and shape indices are used in place of function arguments. The equations ground out in functions edgeScore and arcScore that score individual lattice edges and phrase dependency arcs, respectively.[13] A semiring-generic format is used; for decoding, the semiring "plus" operator ($\oplus$) would be defined as max and the semiring "times" operator ($\otimes$) would be defined as $+$. The entry point when executing the algorithm is to build GOAL, which in turn requires building the other structures.

We use a simple top–down implementation with memoization. Our style of specifying dynamic programming algorithms is similar to weighted deduction, but additionally specifies indices and ranges of iteration, which are useful for a top–down implementation. Top–down dynamic programming avoids the overhead of maintaining a priority queue that is required by bottom–up agenda algorithms (Nederhof 2003; Eisner, Goldlust, and Smith 2005).

The disadvantage of top–down dynamic programming is that wasted work can be done; structures can be built that are never used in any full parse. This problem appears when parsing with context-free grammars, and so the CKY algorithm works bottom–up, starting with the smallest constituents and incrementally building larger ones. This is because context-free grammars may contain rules with only non-terminals. Top–down execution may consider the application of such rules in sequence, producing long derivations of non-terminals that never "ground out" in any symbols in the string. A dependency model, on the other hand, always works directly on words when building items, so a top–down implementation can avoid wasted effort.

However, this situation changes with *lattice* dependency parsing. It is possible for a top–down lattice dependency parser to consider some dependencies that are never used in a full parse. We address this issue in the next section.

6.3 Computational Complexity and Speeding Up Decoding

The lattice parsing algorithm requires $O(E^2V)$ time and $O(E^2 + VE)$ space, where E is the number of edges in the lattice and V is the number of nodes. Typical phrase lattices might easily contain tens of thousands of nodes and edges, making exact search prohibitively expensive for all but the smallest lattices. So we use three techniques to speed up decoding: (1) avoiding construction of items that are **inconsequential** (i.e.,

13 To prevent confusion, we use the term **edge** to refer to a phrase lattice edge and **arc** to refer to a dependency attachment in a dependency tree.

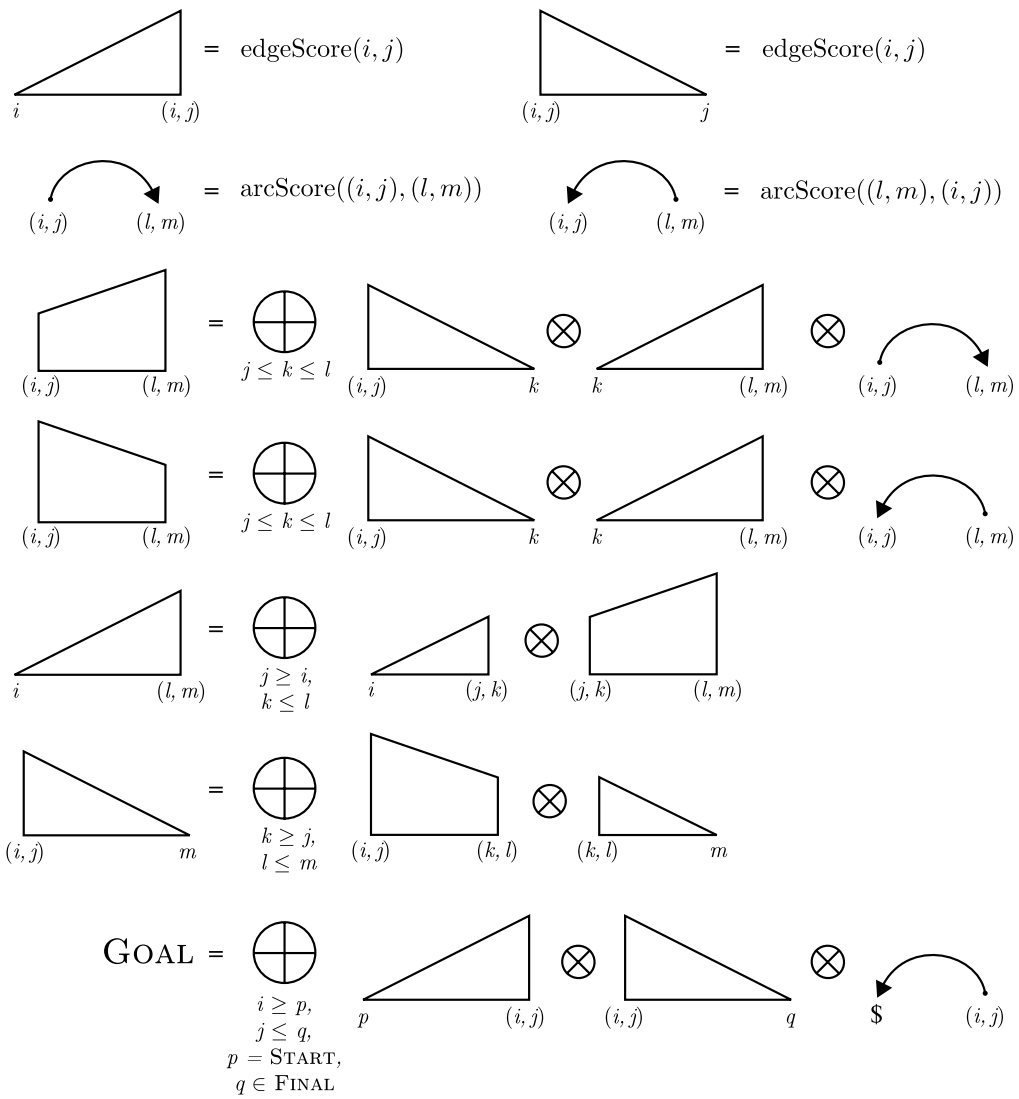


Figure 10
Lattice dependency parsing using an arc-factored dependency model. Lone indices like p and i denote nodes in the lattice, and an ordered pair like (i, j) denotes the lattice edge from node i to node j. START is the single start node in the lattice and FINAL is a set of final nodes. We use edgeScore(i, j) to denote the model score of crossing lattice edge (i, j), which only includes the phrase-based features $\boldsymbol{h}'$. We use arcScore$((i, j), (l, m))$ to denote the score of building the dependency arc from lattice edge (i, j) to its parent (l, m); arcScore only includes the QPD features $\boldsymbol{h}''$.

that could never be contained in a full parse), (2) pruning the lattices, and (3) limiting the maximum length of a phrase dependency.

6.3.1 Avoiding Construction of Inconsequential Items. By design, our phrase lattices impose several types of natural constraints on allowable dependency arcs. For example, each node in the phrase lattice is annotated with a coverage vector—a bit vector indicating

which words in the source sentence have been translated—which implies a topological ordering of the nodes. Once a word in the source sentence has been covered (i.e., translated), it cannot be uncovered later. This can tell us whether certain nodes are unreachable from other nodes. For example, for a three-word source sentence, there cannot exist a directed path from a node with coverage vector $\langle 0, 1, 0 \rangle$ to a node with coverage vector $\langle 0, 0, 1 \rangle$. However, there may or may not be a path from a node with vector $\langle 0, 1, 0 \rangle$ to one with $\langle 0, 1, 1 \rangle$.

Generally, we need an efficient way to determine, for any two nodes in the lattice, whether there exists a path from one to the other. If there is no path, we can avoid wasting time figuring out the best way to build items that would end at the two nodes. To discover this, we use an all-pairs shortest paths algorithm to find the score of the best path between each pair of nodes in the lattice. The algorithm also tells us whether each edge is reachable from each other edge, allowing us to avoid drawing dependencies that will never ground out in a lattice path. We use the Floyd-Warshall algorithm (Floyd 1962). This adds some initial overhead to decoding, but in preliminary experiments we found that it saves more time than it costs. We actually run a modified version of the algorithm that computes the length (in words) of the longest path between any two nodes. If the maximum length between two nodes is ∞, the nodes are unreachable from each other. Before we build an item in the algorithm in Figure 10, we check reachability of the item endpoints and only proceed if one can reach the other.

We modified the algorithm to output maximum lengths because we use the maximum lengths to compute the target-side vine grammar features and constraints, as mentioned in Section 5.2.2. In particular we use a feature $f^{\text{tgt}}_{\text{vine}}$ that is a target-side analog to $f^{\text{src}}_{\text{vine}}$ but using the Floyd-Warshall maximum path lengths in place of the actual lengths.

6.3.2 Lattice Pruning. To reduce phrase lattice sizes, we prune lattice edges using forward–backward pruning (Sixtus and Ortmanns 1999), which has also been used by Tromble et al. (2008). This pruning method computes the max-marginal for each lattice edge, which is the score of the best full path that uses that edge, then prunes edges whose max-marginal is below a certain fraction of the best path score in the lattice. Max-marginals have been used for other coarse-to-fine learning frameworks (Weiss, Sapp, and Taskar 2010) and offer the advantage that the best path in the lattice is preserved during pruning.

We only use the score contribution from the phrase-based features when computing these max-marginals. For each lattice, we use a grid search to find the most liberal threshold that leaves fewer than 2,000 edges in the resulting lattice. As complexity is quadratic in E, forcing E to be less than 2,000 improves runtime substantially. After pruning, the lattices contain more than 10^{16} paths on average and oracle BLEU scores are typically 10–15 points higher than the model-best paths.

6.3.3 Maximum Dependency Lengths. We can easily adapt our vine grammar features to function as hard constraints on allowable dependency trees, as originally done by Eisner and Smith (2005) for monolingual dependency parsing. We use two simple constraints on the maximum length of a phrase dependency used during translation. One constrains the number of source words that are crossed from one aligned source phrase to the other aligned source phrase by the phrase dependency. The other constrains the maximum number of target-side words crossed by any path from one target phrase to the other target phrase in a phrase dependency. During translation, we never build items that would require using dependency arcs that violate these

constraints. In Section 7 we discuss the values we used in our primary experiments and also compare translation quality and decoding speed for several values of these hyperparameters.

6.4 Interaction with Learning

The use of a coarse-to-fine decoding procedure affects how we learn the parameters of our model. We use two separate versions of the phrase-based feature weights: one for lattice generation and one for lattice dependency parsing. This is common with coarse-to-fine strategies—separate instances of coarser parameters are required for each subsequent pass. We first learn parameters for the coarse phrase-based model used to generate phrase lattices. Then, after generating the lattices, we prune them (Section 6.3.2) and use a second round of tuning to learn parameters of the fine model, which includes all phrase-based and QPD feature weights. We initialized the phrase-based feature weights using the default Moses weights. For the QPD features, we initialized the phrase dependency probability feature weights to 0.002 and the weights for all other features to 0.

For tuning, we need the k-best outputs, for which efficient dynamic programming algorithms are available. We use Algorithm 3 from Huang and Chiang (2005), which lazily finds the k best derivations efficiently. In preliminary testing, we found that the k-best lists tended to be dominated by repeated translations with different derivations, so we used the technique presented by Huang, Knight, and Joshi (2006), which finds a *unique* k-best list, returning the highest-scoring derivation for each of k unique translations. This modification requires the maintenance of additional data structures to store all of the previously found string yields for each item built during parsing. This incurs additional overhead but allows us to obtain a far more diverse k-best list given a fixed time and memory budget.

For the first round of tuning, we use RAMPION (Gimpel and Smith 2012b), which performs competitively with minimum error rate training (Och 2003) but is more stable. For training the fine model, however, we found that RAMPION did not lead to substantial improvements over the output of the coarse phrase-based model alone. We found better performance by using a fine learner designed for the k-best reranking setting, in particular the structured support vector machine reranker described by Yadollahpour, Batra, and Shakhnarovich (2013). Though we are doing *lattice* reranking rather than k-best reranking, the learning problem for our fine model is similar to that for k-best reranking in that the decoder is exact (i.e., there is no pruning that could lead to different patterns of search error as the parameters change). That is, phrase lattice generation and pruning (described in Section 6.3.2) only depend on the coarse phrase-based feature weights and the maximum dependency length constraints (described in Section 6.3.3); they do not depend on the fine model parameters.

We now briefly describe how we learn parameters for the fine model via lattice reranking. For simplicity, we will only write the source sentence $\boldsymbol{x}$ and its translation $\boldsymbol{y}$ when describing the reranker and omit the additional input and output variables $\tau_{\boldsymbol{x}}, \pi, \phi, \tau_{\phi}$, and $\boldsymbol{b}$, but they are always present and used for computing features. We assume a tuning set with N source sentences: $\{\boldsymbol{x}_i\}_{i=1}^{N}$. Let $\mathbf{Y}_i^R$ be the set of reference translations for source sentence $\boldsymbol{x}_i$. Let $\mathbf{Y}_i = \{\boldsymbol{y}_i^{(1)} \ldots \boldsymbol{y}_i^{(k)}\}$ denote the set of k candidate translations (outputs of our lattice dependency parsing decoder) for $\boldsymbol{x}_i$. Let $\boldsymbol{y}_i^*$ denote the highest-quality translation in the set, that is, $\boldsymbol{y}_i^* = \operatorname{argmin}_{\boldsymbol{y} \in \mathbf{Y}_i} \ell(\mathbf{Y}_i^R, \boldsymbol{y})$, where $\ell(\mathbf{Y}_i^R, \boldsymbol{y})$ is the negated BLEU+1 score (Lin and Och 2004) of $\boldsymbol{y}$ evaluated against references $\mathbf{Y}_i^R$.

We use the following cost function for sentence i and candidate translation $\boldsymbol{y}$:

$$\mathcal{L}(\mathbf{Y}_i^R, \boldsymbol{y}) = \ell(\mathbf{Y}_i^R, \boldsymbol{y}) - \ell(\mathbf{Y}_i^R, \boldsymbol{y}_i^*) \tag{38}$$

that is, the negated BLEU+1 score of translation $\boldsymbol{y}_i$ relative to that of the best translation ($\boldsymbol{y}_i^*$) in the set.

Yadollahpour, Batra, and Shakhnarovich (2013) formulate the reranking learning problem as an L_2-regularized slack-rescaled structured support vector machine (SSVM; Tsochantaridis et al. 2005). The feature weights $\boldsymbol{\theta}$ for the fine model are learned by solving the following quadratic program:

$$\min_{\boldsymbol{\theta}, \xi_i} \ ||\boldsymbol{\theta}||_2^2 + \lambda \sum_{i \in [N]} \xi_i \tag{39a}$$

$$\text{s.t. } \boldsymbol{\theta}^\top \Big(\boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{y}_i^*) - \boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{y}) \Big) \geq 1 - \frac{\xi_i}{\mathcal{L}(\mathbf{Y}_i^R, \boldsymbol{y})} \tag{39b}$$

$$\xi_i \geq 0, \qquad \forall \boldsymbol{y} \in \mathbf{Y}_i \setminus \boldsymbol{y}_i^*, \tag{39c}$$

In Equation (39b), the violation in the margin ξ_i is scaled by the cost of the translation. Thus if in addition to $\boldsymbol{y}_i^*$ there are other good solutions in the set, the margin for such solutions will not be tightly enforced. On the other hand, the margin between $\boldsymbol{y}_i^*$ and bad solutions will be very strictly enforced. Equation (39) is solved via the 1-slack cutting-plane algorithm of Joachims, Finley, and Yu (2009).[14] During the execution of the cutting-plane algorithm, we compute the tuning set BLEU score with all parameter vector values that are considered. At convergence we return the parameters that led to the highest tuning BLEU score. This helps to bridge the discrepancy between our use of sentence-level BLEU+1 in the loss function and corpus BLEU for final evaluation.

We alternate between generating k-best lists using our lattice parser and solving Equation (39) on the fixed lists, each time pooling all previous iterations' lists. We repeat until the parameters do not change, up to a maximum of 15 iterations. We used k-best lists of size 150 and a fixed, untuned value of $\lambda = 0.1$ for all experiments.

6.5 Comparison to Earlier Work

The decoder described above represents some advances over those presented in earlier papers. Our original decoder was designed for a lexical dependency model; we used lattice dependency parsing on lattices in which each edge contained a single source-target word pair (Gimpel and Smith 2009b). Inference was approximated using cube decoding (Gimpel and Smith 2009a), an algorithm that incorporates non-local features in a way similar to cube pruning (Chiang 2007). After developing our QPD model, we moved to phrase lattices but still approximated inference using an agenda algorithm (Nederhof 2003; Eisner, Goldlust, and Smith 2005) with pre-pruning of dependency edges in a coarse pass (Gimpel and Smith 2011).

14 We used OOQP (Gertz and Wright 2003) to solve the quadratic program in the inner loop, which uses HSL, a collection of Fortran codes for large-scale scientific computation (`www.hsl.rl.ac.uk`).

All decoders used lattice dependency parsing, but our current decoder uses an exact algorithm once two simple approximations are made: the pruning of the lattice and the use of maximum dependency length constraints. Hyperparameters control the severity of these two approximations and the use of an exact parsing algorithm allows us to measure their effects on runtime and accuracy.

7. Experiments and Analysis

We now present experimental results using our QPD model. Because our model extends phrase-based translation models with features on source- and target-side syntactic structures, we can conduct experiments that simulate phrase-based, string-to-tree, and tree-to-tree translation, merely by specifying which feature sets to include. This suggests an additional benefit of using a quasi-synchronous approach for machine translation. By using features rather than constraints, we can simulate a range of translation systems in a single framework, allowing clean experimental comparisons among modeling strategies and combining strengths of diverse approaches.

We describe our experimental setup in Section 7.1 and present our main results in Section 7.2. We measure the impact of using unsupervised parsing in Section 7.2.1 and include feature ablation experiments in Section 7.2.2. We present the results of a manual evaluation in Section 7.3 and give examples. We conclude in Section 7.4 with a runtime analysis of our decoder and show the impact of decoding constraints on speed and translation quality.

7.1 Experimental Setup

In this section we describe details common to the experiments reported in this section. Details about decoding and learning were described in Section 6. Full details about language pairs, data sets, and baseline systems are given in Appendix A and Appendix B. We repeat important details here. We use case-insensitive IBM BLEU (Papineni et al. 2002) for evaluation. To measure significance, we use a paired bootstrap (Koehn 2004) with 100,000 samples ($p \leq 0.05$).

7.1.1 Language Pairs. We consider German→English (DE→EN), Chinese→English (ZH→EN), Urdu→English (UR→EN), and English→Malagasy (EN→MG) translation. These four languages exhibit a range of syntactic divergence from English. They also vary in the availability of resources like parallel data, monolingual target-language data, and treebanks. It is standard practice to evaluate unsupervised parsers on languages that do actually have treebanks, which are used for evaluation. We consider this case as well, comparing supervised parsers for English and Chinese to our unsupervised parsers, but we also want to evaluate our ability to exploit unsupervised parsing for languages that have small or nonexistent treebanks, hence our inclusion of Urdu and Malagasy.

7.1.2 Baselines. We compare our model to several baselines:

- Moses, RAMPION, $S = 200$: This is a standard Moses phrase-based system, trained with RAMPION. The Moses default stack size S of 200 was used during tuning and testing. This is the result one would obtain with an off-the-shelf Moses phrase-based system on these data sets (and trained using RAMPION).

- Moses, RAMPION, $S = 500$: This baseline trains a model in the same way as the previous using $S = 200$, but then uses a larger stack size ($S = 500$) when decoding the test sets. This larger stack size was used for generating phrase lattices for lattice reranking, so it provides a more appropriate baseline for comparing to our model.
- Moses, SSVM reranking: Using phrase lattices generated with the preceding configuration, this baseline uses the SSVM reranker from Section 6.4 on the phrase lattices *with only the Moses phrase-based features*, that is, without any QPD features. This baseline helps to separate out the gains achieved through SSVM reranking and the addition of QPD features.
- Hiero, RAMPION: This is a standard hierarchical phrase-based system (Chiang 2007), as implemented in the Moses toolkit and trained using RAMPION.

We see the three Moses systems as our primary baselines because Moses was used to generate phrase lattices for our system. Our model adds new syntactic structures and features to Moses, but because our decoder use Moses' phrase lattices, our approach can be viewed as rescoring Moses' search space. There are pros and cons to this choice. It lets us build on a strong baseline rather than building a system from scratch. Also, by comparing the third baseline ("Moses, SSVM reranking") to our model, we are able to cleanly measure the contribution of our QPD features. However, Hiero has been shown to perform better than phrase-based systems for certain language pairs (Chiang 2007; Zollmann et al. 2008; Birch, Blunsom, and Osborne 2009), and in these cases Hiero proves to be a strong baseline for our model to beat as well. We note that our QPD features could also be used to rescore Hiero's search space to potentially yield further improvements, but we leave this to future work.

7.1.3 Parsers. Our full QPD model requires parsers for both source and target languages. For each language pair, the target-language parser is only used to parse the target side of the parallel corpus and the source-language parser is only used to parse the source side of the tuning and test sets.

We have access to supervised parsers for Chinese, German, and English, which we used for our experiments. In particular, we used the Stanford parser (Levy and Manning 2003; Rafferty and Manning 2008) for Chinese and German and TurboParser (Martins et al. 2010) for English (see Appendix A for details). The Stanford parser is fundamentally a phrase-structure parser and generates dependency trees via head rules, but we chose it for our experiments for its ease of use and compatibility with the tokenization we used, particularly the Chinese segmentation which we obtained from the Stanford Chinese segmenter.[15]

For Urdu and Malagasy, we turn to unsupervised parsing. To measure the impact of using unsupervised parsers, we also performed experiments in which we replaced supervised parsers for Chinese and English with unsupervised counterparts. We now describe how we trained unsupervised parsers for these four languages.

15 More dependency parsers have been made available by the research community since we began this research and would be natural choices for further experimentation, such as ParZu (Sennrich et al. 2009) for German, the parser model from Bohnet (2010) adapted for German by Seeker and Kuhn (2012), and DuDuPlus (Chen et al. 2012) for Chinese.

The most common approach to unsupervised parsing is to train models on sentences from treebanks (without using the annotated trees, of course) along with their gold standard POS tags. This practice must be changed if we wish to use unsupervised parsing for machine translation, because we do not have gold standard POS tags for our data. Fortunately, Smith (2006) and Spitkovsky et al. (2011) have shown that using automatic POS tags for dependency grammar induction can work as well as or better than gold standard POS tags. For syntax-based translation, Zollmann and Vogel (2011) showed that unsupervised tags could work as well as those from a supervised POS tagger.

For Urdu and Malagasy, we use fully unsupervised POS tagging, using the approach from Berg-Kirkpatrick et al. (2010) with 40 tags. We use the "direct gradient" version optimized by L-BFGS (Liu and Nocedal 1989). For Chinese and English, we use the gold standard POS tags from their respective treebanks for training the parser, then use the Stanford POS tagger (Toutanova et al. 2003) to tag the parallel data, tuning, and test sets. As our dependency parsing model, we use the dependency model with valence (Klein and Manning 2004) initialized with a convex initializer (Gimpel and Smith 2012a). The training procedure is described in Gimpel (2012). Our Chinese and English unsupervised parsers are roughly 30 percentage points worse than supervised parsers in dependency attachment accuracy on standard treebank test sets.

We also compared the supervised and unsupervised parsers to a *uniform-at-random* parser. Well-known algorithms exist for sampling derivations under a context-free grammar for a sentence (Johnson, Griffiths, and Goldwater 2007). These algorithms can be used to sample projective dependency trees by representing a projective dependency grammar using a context-free grammar (Smith 2006; Johnson 2007). We used `cdec` (Dyer et al. 2010) to sample projective dependency trees uniformly at random for each sentence.[16]

We only compared the random parser for *source*-side parsing. Swapping parsers for the target language requires parsing the target side of the parallel corpus, rerunning rule extraction and feature computation with the new parses, and finally re-tuning to learn new feature weights. By contrast, changing the source-side parser only requires re-parsing the source side of the tuning and test sets and re-tuning.

7.2 Results

We now present our main results, shown in Tables 6–9. We see that enlarging the search space results in gains in BLEU, as Moses with stack size 500 typically outperforms Moses with stack size 200. For DE→EN (Table 6), SSVM reranking improves performance even without adding any more features, pushing the numbers close to that of Hiero; and adding our QPD features does not provide any additional improvement.

For the other language pairs, however, we do see significant gains over the Moses baselines. For ZH→EN (Table 7), we see an average gain of 0.5 BLEU over the best Moses baseline when using target syntactic features (TGTTREE), and a total average gain of 0.7 BLEU with the full QPD model (TGTTREE + TREETOTREE). The QPD numbers still lag behind the Hiero results on average, but are statistically indistinguishable from Hiero on two of the three test sets. Our QPD features are able to mostly close the

16 We thank Chris Dyer for implementing this feature in `cdec` for us.

Table 6
%BLEU on tune and test sets for DE→EN translation, comparing the baselines to our QPD model with target syntactic features (TGTTREE) and then also with source syntax (+ TREETOTREE). Here, merely using the additional round of tuning with the SSVM reranker improves the BLEU score to 19.9, which is statistically indistinguishable from the two QPD feature sets. Differences between Hiero and the three 19.9 numbers are at the border of statistical significance; the first two are statistically indistinguishable from Hiero but the third is different at $p = 0.04$.

	German→English		
model	notes	tune	test
Moses	RAMPION, $S = 200$	16.2	19.0
	RAMPION, $S = 500$	16.2	19.2
	SSVM reranking	16.9	19.9
QPD	TGTTREE	17.2	19.9
	TGTTREE + TREETOTREE	17.1	19.9
Hiero	RAMPION	17.1	20.1

Table 7
%BLEU on tune and test sets for ZH→EN translation, showing the contribution of feature sets in our QPD model. Both QPD models are significantly better than the best Moses numbers on test sets 1 and 2, but not on test set 3. The full QPD model is significantly better than the version with only TGTTREE features on test set 1 but statistically indistinguishable on the other two test sets. Hiero is significantly better than the full QPD model on test set 2 but not on the other two.

	Chinese→English					
model	notes	tune	test 1	test 2	test 3	test avg.
Moses	RAMPION, $S = 200$	36.0	35.5	34.3	31.3	33.7
	RAMPION, $S = 500$	36.2	36.1	34.6	31.8	34.2
	SSVM reranking	36.3	36.1	34.6	31.8	34.2
QPD	TGTTREE	37.1	36.8	35.3	32.0	34.7
	TGTTREE + TREETOTREE	37.3	37.2	35.5	31.9	34.9
Hiero	RAMPION	37.3	37.4	36.1	32.1	35.2

performance gap between Moses and Hiero, suggesting that the Moses search space (and even our heavily pruned Moses phrase lattices) has the potential for significant improvements when using the right features.

Results for UR→EN translation are shown in Table 8. Here we only have a supervised parser for English, so the TREETOTREE features are incorporated using our unsupervised Urdu parser. All QPD results are significantly better than all Moses baseline results, but there is no significant difference between the two QPD feature sets. This may be due to our use of unsupervised parsing; perhaps the Urdu parses are too noisy for us to see any benefit from the TREETOTREE features. In Section 7.2.1 we measure the impact of using unsupervised parsing for ZH→EN translation. Hiero still significantly outperforms the QPD model, although we have halfway closed the gap between Moses and Hiero.

Table 8
%BLEU on tune and test sets for UR→EN translation, using our unsupervised Urdu parser to incorporate source syntactic features. The two QPD rows are statistically indistinguishable on both test sets. Both are significantly better than all Moses results, but Hiero is significantly better than all others.

Urdu→English					
model	notes	tune	test 1	test 2	test avg.
Moses	RAMPION, $S = 200$	24.6	24.6	24.5	24.5
	RAMPION, $S = 500$	24.7	24.8	24.9	24.8
	SSVM reranking	24.9	24.4	24.7	24.6
QPD	TGTTREE	25.8	25.4	25.5	25.4
	TGTTREE + (unsup.) TREETOTREE	25.8	25.4	25.6	25.5
Hiero	RAMPION	25.7	26.4	26.7	26.6

Table 9
%BLEU on tune and test sets for EN→MG translation, using a supervised English parser and an unsupervised Malagasy parser. The 15.6 BLEU reached by the full QPD model is statistically significantly better than all other results on the test set. All other test set numbers are statistically indistinguishable.

English→Malagasy			
model	notes	tune	test
Moses	RAMPION, $S = 200$	17.6	15.1
	RAMPION, $S = 500$	17.8	15.1
	SSVM reranking	17.8	15.1
QPD	(unsup.) TGTTREE	17.6	15.2
	(unsup.) TGTTREE + TREETOTREE	17.9	15.6
Hiero	RAMPION	17.4	15.0

For EN→MG translation (Table 9), we see significant gains in BLEU over both Moses and Hiero when using the full QPD model.[17] We used unsupervised parsing to incorporate TGTTREE features but we only see a statistically significant improvement when we add TREETOTREE features, which use a supervised English parser.

7.2.1 Impact of Unsupervised Parsing. Table 10 shows results when comparing parsers for ZH→EN translation. We pair supervised and unsupervised parsers for English and Chinese. The final row shows the Moses BLEU scores for comparison.

17 For the EN→MG experiments, we modified our initialization procedure for the QPD feature weights. When using the same initialization as the other language pairs (setting QPD probability feature weights to 0.002 and all other QPD weights to 0), we found that SSVM reranking did not find any higher BLEU score in the initial k-best lists than the 1-best translations for all sentences. So we multiplied the initial QPD weights by 10 in an effort to inject more diversity in the initial k-best lists.

Table 10
%BLEU on tune and test sets when comparing parsers for ZH→EN translation. QPD uses all features, including TGTTREE and TREETOTREE. The table first pairs supervised English parsing with supervised, unsupervised, and random Chinese parsing, then pairs unsupervised English parsing with supervised and unsupervised Chinese parsing. † = significantly better than sup/sup, ∗ = significantly worse than sup/sup.

	EN parser	ZH parser	tune %BLEU	test 1 %BLEU	test 2 %BLEU	test 3 %BLEU	avg. test %BLEU
QPD	sup.	sup.	37.3	37.2	35.5	31.9	34.9
		unsup.	37.2	37.0	35.8†	31.8	34.9
		random	37.1	36.5∗	35.2	31.6∗	34.4
	unsup.	sup.	37.2	37.1	35.3	31.7∗	34.7
		unsup.	37.2	36.8∗	35.3	31.5∗	34.5
Moses, RAMPION, $S = 500$			36.2	36.1∗	34.6∗	31.8	34.2

When using supervised English parsing, we find that using our unsupervised Chinese parser in place of the Stanford parser leads to the same average test set BLEU score. When instead using random Chinese parses, we see a significant drop on two of the three test sets and an average decrease of 0.5 BLEU. When pairing unsupervised English parsing with supervised Chinese parsing, we see an average drop of just 0.2 BLEU compared to the fully supervised case. When both parsers are unsupervised, BLEU scores drop further but are still above the best Moses baseline on average.

One idea that we have not explored is to parse our parallel corpus using each parser (unsupervised and supervised), then extract rules consistent with any of the parses. This might give us some of the benefits of forest-based rule extraction, which has frequently been shown to improve translation quality (Liu et al. 2007; Mi, Huang, and Liu 2008; Mi and Huang 2008). Similarly, because we train systems for several language pairs, we could pool the rules extracted from all parallel corpora for computing target-syntactic features. For example, adding the English phrase dependency rules from the DE→EN corpus could improve performance of our ZH→EN and UR→EN systems. Moving beyond translation, we could use the pool of extracted rules from all systems (and using all parsers) to build *monolingual* phrase dependency parsers for use in other applications (Wu et al. 2009).

7.2.2 Feature Ablation. We performed feature ablation experiments for UR→EN translation, shown in Table 11. Starting with TGTTREE features, which consist of word (WORD), cluster (CLUST), and configuration (CFG) feature sets, we alternately removed each of the three. We find only a small (and statistically insignificant) drop in BLEU when omitting word features, but a larger drop when omitting word cluster features. This may be due to the small size of our training data for UR→EN (approximately 1 million words of parallel text). With limited training data, it is not surprising that unlexicalized features like the cluster and configuration features would show a stronger effect than the lexicalized features.

7.3 Human Evaluation

We focused on UR→EN and ZH→EN translation for our manual evaluation, as these language pairs showed the largest gains in BLEU when using our QPD model. We

Table 11
Feature ablation experiments for UR→EN translation with string-to-tree features, showing the drop in BLEU when separately removing word (WORD), cluster (CLUST), and configuration (CFG) feature sets. $*$ = significantly worse than TGTTREE. Removing word features causes no significant difference. Removing cluster features results in a significant difference on both test sets, and removing configuration features results in a significant difference on test 2 only.

Urdu→English					
model	notes	tune	test 1	test 2	test avg. (Δ)
Moses	SSVM reranking	24.9	24.4	24.7	24.6
QPD	TGTTREE = WORD + CLUST + CFG	25.8	25.4	25.5	25.4
	TGTTREE – WORD	25.6	25.0	25.5	25.2 (-0.2)
	TGTTREE – CLUST	25.4	24.8*	24.9*	24.9 (-0.5)
	TGTTREE – CFG	25.1	25.1	25.0*	25.0 (-0.4)

began by performing a human evaluation using Amazon Mechanical Turk (MTurk) in order to validate the BLEU differences against human preference judgments and to identify translations that were consistently judged better under each model for follow-up manual evaluation.

7.3.1 Procedure. We first removed sentences with unknown words, as we feared they would only confuse judges.[18] We then randomly selected 500 sentences from UR→EN test 2 and 500 from the concatenation of ZH→EN test 1 and test 2. For each of the 1,000 sentences, we chose a single reference translation from among the four references to show to judges.[19] All text was detokenized. Judges were shown the reference translation, the translation from the Moses system with SSVM reranking, and the translation from our QPD system with the full feature set. We randomized the order in which the two machine translations were presented. Judges were asked to select which translation was closer in meaning to the reference; alternatively, they could indicate that they were of the same quality. We obtained judgments like these from three judges for each of the 1,000 sentences.

7.3.2 Results and Analysis. Table 12 shows the results of our MTurk experiments. If a sentence was judged to be translated better by one system more often than the other, it was counted as a victory for that system. The QPD translations for 40–43% of the sentences were preferred over Moses, but for 28–33% of the sentences, the reverse was true.

We can use these judgments to study when and how our system improves over Moses, and also when Moses still performs better. For a follow-up manual evaluation, we looked at ZH→EN sentences for which all three judges selected either Moses or the QPD model; these should be the clearest examples of success for each system. In

18 Although this filtering step may introduce bias, we confirmed that the system differences in BLEU were similar whether looking at sentences with unknown words, those without unknown words, or all sentences.

19 For UR→EN test 2 and ZH→EN test 2, we chose the first reference set from the four provided. For ZH→EN test 1, we instead chose the second reference set because its average length was closer to the average across the four reference sets.

Table 12
Results of human evaluation performed via Amazon Mechanical Turk. The percentages represent the portion of sentences for which one system had more preference judgments than the other system. If a sentence had an equal number of judgments for the two systems, it was counted in the final row ("neither preferred").

	% of sentences preferred	
	ZH→EN	UR→EN
Moses, SSVM reranking	33.4%	28.6%
QPD, TGTTREE + TREETOTREE	40.6%	42.8%
neither preferred	26.0%	28.6%

looking at these sentences, we attempted to categorize the primary reasons why all three judges would have preferred one system's output over the other. We began with two broad categories of improvement: word choice and word order. We divided word choice improvements into two subcategories: those involving verbs and those involving words other than verbs. The reason we made this distinction is because some differences in non-verb translation are not as crucial for understanding a sentence as differences in verb translation or word order. Anecdotally, we observed that when one sentence has a better verb translation and the other has a better preposition translation, judges tend to prefer the translation with the better verb. We noted some sentences that fit multiple categories, but in our analysis we chose a single category that we deemed to be the most important factor in the judges' decisions.

Of the 26 sentences for which Moses output was preferred unanimously, we agreed with the consensus on 25 and found that 19 of these improved due to better word choice, most frequently (13 out of 19) for words other than verbs. Only 6 of the 25 were determined to be preferred due to word order. The top section of Table 13 shows representative examples when Moses' translations were unanimously preferred. Moses handles prepositions and other function words better than the QPD model in these examples. This may occur due to the reliance of phrase-based systems upon strong n-gram language models to ensure local fluency. The QPD model uses all of Moses' features, including the same n-gram language model, but it adds many other features that score longer-distance word order and may be overwhelming the n-gram model in certain cases.

For the 44 sentences for which QPD output was unanimously preferred, we agreed with the judges on 42. Of these, we found that 15 had improved word order, 14 had improvements in verb word choice, and 13 had improved word choice for non-verbs. So the QPD model's improvements were due to word order on 36% of unanimous sentences, compared with Moses' 24%, suggesting that the QPD model's strength is in improving word order. The lower section of Table 13 shows representative examples. Consider the final example in the table. The Moses translation has better local fluency, but mixes words across clauses, confusing the meaning. The QPD translation has two local disfluencies ("because law-abiding citizen hopes to" and "don't need to fear will attack"), but has no erroneous word reordering across clause boundaries.

7.4 Decoding Speed

So far we have reported BLEU scores for various feature sets and parsers, but we have not discussed decoding speed. BLEU improvements may not be worth substantial

reductions in translation speed. In this section we report decoding speeds and BLEU scores for UR→EN translation as pruning thresholds are varied. Our lattice dependency parsing decoding algorithm is exact, but two pruning stages precede lattice parsing, as discussed in Section 6.3: (1) pruning the phrase lattices based on the phrase-based model

Table 13
Examples from manual evaluation of ZH→EN output, selected from sentences with unanimous preference judgments for Moses (top) and our QPD model (bottom). **R** = reference; **M** = Moses, SSVM reranking; **Q** = QPD, TGTTREE + TREETOTREE.

Moses preferred:

R the purpose of cheney's trip is to win supports of middle east countries in case of a u.s. attack on iraq.
M cheney's trip is to lobby the middle east countries to support the us attack on iraq.
Q cheney's trip was to lobby the middle east countries support us attacks on iraq.

R he said, "after this conference, our position is clear to all."
M he said: "this time after the talks, our position is well known."
Q he said: "this round of talks, our position is well known."

R ukraine condemned us for interfering its internal affairs.
M ukraine condemned the united states for interfering in its internal affairs.
Q ukraine condemned the united states of interfering in its internal affairs.

R former serbian interior minister, vlajko stojiljkovic, attempts to commit suicide.
M yugoslavia, a former serbian interior minister tried to commit suicide.
Q interior minister tried to commit suicide before the serbian republic of yugoslavia.

R arafat died in a french hospital of an unknown illness on november 11.
M arafat died in a french hospital on nov. 11, due to an unidentified disease.
Q arafat died on nov. 11 in a french hospital, an unidentified disease.

QPD preferred:

R bush: nato must forge a new relationship with russia
M bush said that the establishment of a new type of relationship between russia and nato must
Q bush says nato must build new relations with russia

R that would portend increased flows of glacial ice from antarctica and a rise in global sea levels, endangering low-lying land areas.
M this bodes well for the antarctica ice flow will increase, the global sea level will rise in the lowlands, will cause harm to the region.
Q this indicates that the antarctica's ice flow will increase, global sea level will rise, and will cause harm to the lowlands region.

R this is the first sandstorm in north korea since spring this year.
M this is the first since the spring of this year the sandstorms.
Q this is the first sandstorm of the korean experience since spring this year.

R "we really have to tackle the problem of street-crime. law-abiding citizens want to feel safe when they walk on the street. they want a peaceful life and untroubled by attack," he said.
M he said: "we really should rectify the order because of law-abiding citizens, in the hope of peace and security on the streets and on the living, not to be afraid of attacks."
Q he said: "we really should rectify the order in the streets, because law-abiding citizen hopes to secure a safe life on the streets, and don't need to fear will attack."

scores, and (2) pruning the search space deterministically based on source- and target-side limits on dependency lengths. In this section, we measure the impact of the latter type of pruning only.[20]

We vary maximum dependency lengths and we report BLEU scores and decoding speeds. We find that we can set these limits to be relatively strict and get similar BLEU scores in less time. In all previous experiments, we used a source-side limit ω_x of 15 and a target-side limit ω_y of 20. That is, all target-side phrase dependencies may cover a maximum of 20 words in the target sentence, and the number of words between the aligned source phrases can be at most 15. We often use a larger value of ω_y because it is constraining an *upper bound* on the number of words crossed in the translation, whereas ω_x constraints the *exact* number of source words crossed by a dependency (see Section 6.3.3 for details).

For timing experiments, we ran a single decoding thread on a Sun Fire X2200 M2 x64 server with two 2.6-GHz dual-core CPUs. Decoding during *tuning* is time-consuming, because we generate unique 150-best lists for each iteration, so we only use two max dependency length settings for tuning. But given trained models, finding the 1-best output on the test data is much faster. So we experimented with more pruning settings for decoding. Table 14 shows our results. The upper table reminds us of the baseline BLEU scores. The lower table shows what happens when we train with two pruning settings: ($\omega_x = 10, \omega_y = 15$) and ($\omega_x = 15, \omega_y = 20$), and test with many others.

The times reported only include the time required for running the Floyd-Warshall algorithm on the lattice and performing lattice dependency parsing. We use the Moses decoder for lattice generation; this typically takes only slightly longer than ordinary decoding, which is generally in the range of a couple seconds per sentence, depending on how the phrase table and language model are accessed. The average time required to run the Floyd-Warshall algorithm on the lattices is approximately 0.8 seconds per sentence, so it begins to dominate the total time as the pruning thresholds go below (5, 5). The earlier numbers in this section used ($\omega_x = 15, \omega_y = 20$) for both tuning and testing, which causes test-time decoding to take approximately 6 seconds per sentence, as shown in the table. We can see that we can use stricter constraints during test-time decoding only (e.g., ($\omega_x = 5, \omega_y = 10$)) and speed this up by a factor of 3 while only losing 0.1 BLEU. The only severe drops in BLEU appear when using thresholds below (5, 5).

8. Conclusion and Future Work

We presented a new approach to machine translation that combines phrases, dependency syntax, and quasi-synchronous tree-to-tree relationships. We introduced several categories of features for dependency-based translation, including string-to-tree and tree-to-tree features. We proposed lattice dependency parsing to solve the decoding problem and presented ways to speed up the search and prune the search space. We presented experimental results on seven test sets across four language pairs, finding statistically significant improvements over strong phrase-based baselines on five of the seven. Manual inspection reveals improvement in the translation of verbs, an important component in preserving the meaning of the source text. We showed that unsupervised

20 Ideally we could also measure the impact of pruning the phrase lattices to various sizes, but this would require the time-consuming process of filtering our phrase dependency tables for each lattice size, so we have not yet tested the effect of this pruning systematically.

Table 14
%BLEU on tune and test sets for UR→EN translation, comparing several settings for maximum dependency lengths in the decoder (ω_x is for the source side and ω_y is for the target side). The upper table shows Moses BLEU scores for comparison. The lower table compares two max dependency length settings during tuning, and several for decoding on the test sets, showing both BLEU scores and average decoding times per sentence. See text for discussion.

	tune %BLEU	test 1 %BLEU	test 2 %BLEU	avg. test %BLEU
Moses, SSVM reranking	24.9	24.4	24.7	24.6

	tune (ω_x, ω_y)	tune %BLEU	test (ω_x, ω_y)	test 1 %BLEU	test 2 %BLEU	avg. test %BLEU	time (sec./sent.)
QPD	(10, 15)	25.9	(3, 3)	21.7	22.3	22.0	1.11
			(3, 5)	23.2	23.5	23.3	1.28
			(5, 5)	24.9	24.7	24.8	1.41
			(5, 10)	25.2	25.0	25.1	2.09
			(10, 10)	25.4	25.3	25.3	3.01
			(10, 15)	25.3	25.4	25.4	4.00
			(15, 20)	25.5	25.5	25.5	6.15
			(20, 20)	25.6	25.5	25.6	7.18
			(20, 25)	25.5	25.5	25.5	7.83
	(15, 20)	25.8	(3, 3)	22.2	22.8	22.5	1.11
			(3, 5)	23.2	24.0	23.6	1.28
			(5, 5)	25.2	25.2	25.2	1.41
			(5, 10)	25.3	25.4	25.4	2.08
			(10, 10)	25.2	25.5	25.4	3.01
			(10, 15)	25.4	25.6	25.5	4.02
			(15, 20)	25.4	25.6	25.5	6.07
			(20, 20)	25.4	25.6	25.5	7.16
			(20, 25)	25.4	25.6	25.5	7.96

dependency parsing can be used effectively within a tree-to-tree translation system, enabling the use of our system for low-resource languages like Urdu and Malagasy. This result offers promise for researchers to apply syntactic translation models to languages for which we do not have manually annotated corpora.

There are many directions for future work. Unsupervised learning of syntax can be improved if parallel text is available and we have a parser for one of the languages: The parallel text can be word-aligned and the annotations can be projected across the word alignments (Yarowsky and Ngai 2001; Yarowsky, Ngai, and Wicentoswki 2001). The projected parses can be improved by applying manually written rules (Hwa et al. 2005) or modeling the noisy projection process (Ganchev, Gillenwater, and Taskar 2009; Smith and Eisner 2009). If we do not have parsers for either language, grammar induction models have been developed to exploit parallel text without using any annotations on either side (Kuhn 2004; Snyder, Naseem, and Barzilay 2009). Techniques are also available for grammar induction using treebanks in different languages that are not built on parallel data (Cohen, Das, and Smith 2011).

Researchers have recently begun to target learning of parsers specifically for applications like machine translation. Hall et al. (2011) developed a framework to train supervised parsers for use in particular applications by optimizing arbitrary evaluation metrics; Katz-Brown et al. (2011) used this framework to train a parser for reordering

in machine translation. Relatedly, DeNero and Uszkoreit (2011) tailored unsupervised learning of syntactic structure in parallel text to target reordering phenomena.

In addition, we may not need full monolingual syntactic parses to obtain the benefits of syntax-based translation modeling. Indeed, the widely used hierarchical phrase-based model of Chiang (2005) induces a synchronous grammar from parallel text without any linguistic annotations. Zollmann and Vogel (2011) and Zollmann (2011) showed that using a supervised POS tagger to label these synchronous rules can improve performance up to the level of a model that uses a supervised full syntactic parser. They further showed that unsupervised POS taggers could be effectively used in place of supervised taggers. These results suggest that it may be fruitful to explore the use of simpler annotation tools such as POS taggers, whether supervised or unsupervised, in order to apply syntax-based translation to new language pairs.

Appendix A. Language Pairs

We consider four language pairs in this article, two for which large amounts of parallel data are available and two involving low-resource languages. The large-data language pairs we consider are Chinese→English (ZH→EN) and German→English (DE→EN). The two low-resource language pairs are Urdu→English (UR→EN) and English→Malagasy (EN→MG).

For all language pairs, English text was parsed using TurboParser version 0.1 (Martins et al. 2010). We used a second-order model with sibling and grandparent features that was trained to maximize conditional log-likelihood.

The following sections describe the procedures used to prepare the data for each language pair. The line and token counts are summarized in Tables A.1–A.3.

Chinese→English. For ZH→EN, we used 303k sentence pairs from the FBIS corpus (LDC2003E14). We segmented the Chinese data using the Stanford Chinese segmenter (Chang, Galley, and Manning 2008) in "CTB" mode, giving us 7.9M Chinese

Table A.1
Statistics of data used for rule extraction and feature computation.

	lines	source tokens	target tokens
ZH→EN	302,996	7,984,637	9,350,506
DE→EN	1,010,466	23,154,642	24,044,528
UR→EN	165,159	1,169,367	1,083,807
EN→MG	83,670	1,500,922	1,686,022

Table A.2
Statistics of data used for tuning. The numbers of target tokens are averages across four reference translations for ZH→EN and UR→EN, rounded to the nearest token.

	lines	source tokens	target tokens
ZH→EN	919	24,152	28,870
DE→EN	1,300	29,791	31,318
UR→EN	882	18,004	16,606
EN→MG	1,359	28,408	32,682

Table A.3
Test data statistics. The numbers of target tokens are averages across four reference translations for ZH→EN and UR→EN, rounded to the nearest token.

	test 1			test 2			test 3		
	lines	source tokens	target tokens	lines	source tokens	target tokens	lines	source tokens	target tokens
ZH→EN	878	22,708	26,877	1,082	29,956	35,227	1,664	38,787	48,169
DE→EN	2,525	62,699	65,595		N/A			N/A	
UR→EN	883	21,659	19,575	1,792	42,082	39,889		N/A	
EN→MG	1,133	24,362	28,301		N/A			N/A	

tokens and 9.4M English tokens. For tuning and testing, we used MT03 ("tune"), MT02 ("test 1"), MT05 ("test 2"), and MT06 ("test 3"). The Chinese text was parsed using the Stanford parser (Levy and Manning 2003).

German→English. We started with the Europarl corpus provided for the WMT12 shared task. We tokenized both sides, filtered sentences with more than 50 words, and downcased the text. We then discarded every other sentence, beginning with the second, leaving half of the corpus remaining. We did this to speed our experiment cycle. The corpus still has about 850k sentence pairs. We did the same processing with the news commentary corpus, but did not discard half of the sentences. There were about 150k news commentary sentences, giving us a total of about 1M lines of DE→EN parallel training data. For tuning, we used the first 1,300 sentences from the 2008 2,051-sentence test set ("tune"). For testing, we used the 2009 test set ("test 1"). The tuning/test sets are from the newswire domain. The German text was parsed using the factored model in the Stanford parser (Rafferty and Manning 2008).

Urdu→English. For UR→EN, we used parallel data from the NIST MT08 evaluation. Although there are 165,159 lines of parallel data, there are many dictionary and otherwise short entries, so it is close to an order of magnitude smaller than ZH→EN. We used half of the documents (882 sentences) from the MT08 test set for tuning ("tune"). We used the remaining half for one test set ("test 1") and MT09 as a second test set ("test 2"). The Urdu text was parsed using an unsupervised dependency parser as described in Section 7.1.3.

English→Malagasy. For EN→MG translation, we used data obtained from the Global Voices weblogging community (`http://globalvoicesonline.org`), prepared by Victor Chahuneau.[21] We used release 12.06 along with its recommended training, development (tuning), and test set. Like Urdu, the Malagasy text was parsed using an unsupervised dependency parser as described in Section 7.1.3.

Appendix B. Experimental Details

Appendix A contains details about the data sets used in our experiments. Other experimental details are given here.

21 The data are publicly available at `http://www.ark.cs.cmu.edu/global-voices/`.

Translation Models. For phrase-based models, we used the Moses machine translation toolkit (Koehn et al. 2007). We mostly used default settings and features, including the default lexicalized reordering model. Word alignment was performed using GIZA++ (Och and Ney 2003) in both directions, the `grow-diag-final-and` heuristic was used to symmetrize the alignments, and a max phrase length of 7 was used for phrase extraction. The only exception to the defaults was setting the distortion limit to 10 in all experiments.

Language Models. Language models were trained using the target side of the parallel corpus in each case augmented with 24,760,743 lines (601,052,087 tokens) of randomly selected sentences from the Gigaword v4 corpus (excluding the *New York Times* and *Los Angeles Times*). The minimum count cutoff for unigrams, bigrams, and trigrams was one and the cutoff for fourgrams and fivegrams was three. Language models were estimated using the SRI Language Modeling toolkit (Stolcke 2002) with modified Kneser-Ney smoothing (Chen and Goodman 1998). Language model inference was performed using KenLM (Heafield 2011) within Moses.

For EN→MG, we estimated a 5-gram language model using only the target side of the parallel corpus, which contained 89,107 lines with 2,031,814 tokens. We did not use any additional Malagasy data for estimating the EN→MG language models in order to explore a scenario in which target-language text is limited or expensive to obtain.

Word Clustering. Brown clusters (Brown et al. 1992) were generated using code provided by Liang (2005). For each language pair, 100 word clusters were generated for the target language. The implementation allows the use of a token count cutoff, which causes the algorithm to only cluster words appearing more times than the cutoff. When the clusters are used, all words with counts below the cutoff are assigned a special "unknown word" cluster. So in practice, if a clustering with 100 clusters is generated, there are 101 clusters used when the clusters are applied.

For ZH→EN, DE→EN, and UR→EN, the target side of the parallel data was used along with 412,000 lines of randomly selected Gigaword data comprising 10,001,839 words. This data was a subset of the Gigaword data used for language modeling. The count cutoff was 2. For EN→MG, only the target side of the parallel corpus was used. The count cutoff was 1. In all cases, the data was tokenized and downcased prior to cluster generation.

Acknowledgments

We thank the anonymous reviewers as well as Dhruv Batra, Jaime Carbonell, David Chiang, Shay Cohen, Dipanjan Das, Chris Dyer, Jason Eisner, Alon Lavie, André Martins, Greg Shakhnarovich, David Smith, Stephan Vogel, and Eric Xing. This research was supported in part by the National Science Foundation through grant IIS-0844507, the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number W911NF-10-1-0533, and Sandia National Laboratories (fellowship to K. Gimpel).

References

Aho, A. V. and J. D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37–56.

Ambati, V. and A. Lavie. 2008. Improving syntax driven translation models by re-structuring divergent and non-isomorphic parse tree structures. In *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas*, pages 235–244, Waikiki, HI.

Berg-Kirkpatrick, T., A. Bouchard-Côté, J. DeNero, and D. Klein. 2010. Painless

unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 582–590, Los Angeles, CA.

Birch, A., P. Blunsom, and M. Osborne. 2009. A quantitative analysis of reordering phenomena. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 197–205, Athens.

Blunsom, P. and T. Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1,204–1,213, Cambridge, MA.

Bohnet, B. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing.

Brown, P. F., P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Buchholz, S. and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City.

Carreras, X. and M. Collins. 2009. Non-projective parsing for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 200–209, Singapore.

Chang, P., M. Galley, and C. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 224–232, Columbus, OH.

Charniak, E. and M. Johnson. 2005. Coarse-to-fine *n*-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, MI.

Chen, S. and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report 10-98, Harvard University.

Chen, W., J. Kazama, K. Uchimoto, and K. Torisawa. 2012. Exploiting subtrees in auto-parsed data to improve dependency parsing. *Computational Intelligence*, 28(3):426–451.

Chiang, D. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, MI.

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

Chiang, D. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1,443–1,452, Uppsala.

Christodoulopoulos, C., S. Goldwater, and M. Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584, Cambridge, MA.

Cohen, S. B. 2011. *Computational Learning of Probabilistic Grammars in the Unsupervised Setting*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

Cohen, S. B., D. Das, and N. A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 50–61, Edinburgh.

Cowan, B., I. Kučerová, and M. Collins. 2006. A discriminative model for tree-to-tree translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 232–241, Sydney.

Crego, J. M. and F. Yvon. 2009. Gappy translation units under left-to-right SMT decoding. In *Proceedings of the Meeting of the European Association for Machine Translation (EAMT)*, pages 66–73, Barcelona.

Das, D. and N. A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 468–476, Suntec.

DeNero, J. and J. Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Edinburgh.

Ding, Y. and M. Palmer. 2005. Machine translation using probabilistic

synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 541–548, Ann Arbor, MI.

Dorr, B. J. 1994. Machine translation divergences: a formal description and proposed solution. *Computational Linguistics*, 20(4):597–633.

Dyer, C., A. Lopez, J. Ganitkevitch, J. Weese, F. Ture, P. Blunsom, H. Setiawan, V. Eidelman, and P. Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12, Uppsala.

Eisner, J. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen.

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

Eisner, J., E. Goldlust, and N. A. Smith. 2005. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 281–290, Vancouver.

Eisner, J. and N. A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proceedings of IWPT*, pages 30–41, Vancouver.

Floyd, R. W. 1962. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345.

Fox, H. J. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 304–311, Philadelphia, PA.

Galley, M. and C. D. Manning. 2009. Quadratic-time dependency parsing for machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 773–781, Suntec.

Galley, M. and C. D. Manning. 2010. Accurate non-hierarchical phrase-based translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 966–974, Los Angeles, CA.

Ganchev, K., J. Gillenwater, and B. Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 369–377, Suntec.

Gao, Y., P. Koehn, and A. Birch. 2011. Soft dependency constraints for reordering in hierarchical phrase-based translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 857–868, Edinburgh.

Gertz, E. M. and S. J. Wright. 2003. Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software*, 29(1):58–81.

Gildea, D. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 80–87, Sapporo.

Gimpel, K. 2012. *Discriminative Feature-Rich Modeling for Syntax-Based Machine Translation*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

Gimpel, K. and N. A. Smith. 2008. Rich source-side context for statistical machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 9–17, Columbus, OH.

Gimpel, K. and N. A. Smith. 2009a. Cube summing, approximate inference with non-local features, and dynamic programming without semirings. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 318–326, Athens.

Gimpel, K. and N. A. Smith. 2009b. Feature-rich translation by quasi-synchronous lattice parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 219–228, Singapore.

Gimpel, K. and N. A. Smith. 2011. Quasi-synchronous phrase dependency grammars for machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 474–485, Edinburgh.

Gimpel, K. and N. A. Smith. 2012a. Concavity and initialization for unsupervised dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 577–581, Montréal.

Gimpel, K. and N. A. Smith. 2012b. Structured ramp loss minimization for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231, Montréal.

Hall, K., R. McDonald, J. Katz-Brown, and M. Ringgaard. 2011. Training dependency parsers by jointly optimizing multiple objectives. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1,489–1,499, Edinburgh.

Hanneman, G. and A. Lavie. 2011. Automatic category label coarsening for syntax-based machine translation. In *Proceedings of Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 98–106, Portland, OR.

Heafield, K. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh.

Huang, L. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, OH.

Huang, L. and D. Chiang. 2005. Better *k*-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64, Vancouver.

Huang, L., K. Knight, and A. Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the Association for Machine Translation in the Americas*, pages 66–73, Cambridge, MA.

Hunter, T. and P. Resnik. 2010. Exploiting syntactic relationships in a phrase-based decoder: An exploration. *Machine Translation*, 24(2):123–140.

Hwa, R., P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Journal of Natural Language Engineering*, 11(3):311–25.

Joachims, T., T. Finley, and Chun-Nam Yu. 2009. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59.

Johnson, M. 2007. Transforming projective bilexical dependency grammars into efficiently-parsable CFGs with unfold-fold. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 168–175, Prague.

Johnson, M., T. Griffiths, and S. Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, NY.

Katz-Brown, J., S. Petrov, R. McDonald, F. Och, D. Talbot, H. Ichikawa, M. Seno, and H. Kazawa. 2011. Training a parser for machine translation reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 183–192, Edinburgh.

Klein, D. and C. D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, PA.

Klein, D. and C. D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 3–10, Vancouver.

Klein, D. and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 478–485, Barcelona.

Koehn, P. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395, Barcelona.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague.

Koehn, P., F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 48–54, Edmonton.

Kübler, S., R. McDonald, and J. Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool.

Kuhn, J. 2004. Experiments in parallel-text based grammar induction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 470–477, Barcelona.

Levy, R. and C. D. Manning. 2003. Is it harder to parse Chinese, or the Chinese treebank? In *Proceedings of the 41st Annual Meeting of*

the Association for Computational Linguistics, pages 439–446, Sapporo.
Li, Z. and S. Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of the ACL-08: HLT Second Workshop on Syntax and Structure in Statistical Translation (SSST-2)*, pages 10–18, Columbus, OH.
Li, Z., T. Liu, and W. Che. 2012. Exploiting multiple treebanks for parsing with quasi-synchronous grammars. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–684, Jeju Island.
Liang, P. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.
Lin, C. and F. J. Och. 2004. Orange: A method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of Coling 2004*, pages 501–507, Geneva.
Lin, D. 2004. A path-based transfer model for machine translation. In *Proceedings of Coling 2004*, pages 625–630, Geneva.
Liu, D. C. and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528.
Liu, Y., Y. Huang, Q. Liu, and S. Lin. 2007. Forest-to-string statistical translation rules. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 704–711, Prague.
Liu, Y., Y. Lü, and Q. Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 558–566, Suntec.
Marcus, M. P., B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.
Martins, A. F. T., N. A. Smith, and E. P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 342–350, Suntec.
Martins, A. F. T., N. A. Smith, E. P. Xing, P. M. Q. Aguiar, and M. A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44, Cambridge, MA.
Melamed, I. D. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 79–86, Edmonton.
Mi, H. and L. Huang. 2008. Forest-based translation rule extraction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 206–214, Honolulu, HI.
Mi, H., L. Huang, and Q. Liu. 2008. Forest-based translation. In *Proceedings of ACL-08: HLT*, pages 192–199, Columbus, OH.
Naseem, T., H. Chen, R. Barzilay, and M. Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1,234–1,244, Cambridge, MA.
Nederhof, M.-J. 2003. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–143.
Nivre, J., J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague.
Och, F. J. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo.
Och, F. J. and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.
Papineni, K., S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA.
Park, J. H., W. B. Croft, and D. A. Smith. 2011. A quasi-synchronous dependence model for information retrieval. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 17–26, Glasgow.

Petrov, S. 2009. *Coarse-to-Fine Natural Language Processing*. Ph.D. thesis, University of California at Berkeley.

Quirk, C., A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 271–279, Ann Arbor, MI.

Rafferty, A. N. and C. D. Manning. 2008. Parsing three German treebanks: Lexicalized and unlexicalized baselines. In *Proceedings of the Workshop on Parsing German*, pages 40–46, Columbus, OH.

Riezler, S. and J. T. Maxwell III. 2006. Grammatical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 248–255, New York City.

Seeker, W. and J. Kuhn. 2012. Making ellipses explicit in dependency conversion for a German treebank. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 3,132–3,139, Istanbul.

Sennrich, R., G. Schneider, M. Volk, and M. Warin. 2009. A new hybrid dependency parser for German. In *Proceedings of the German Society for Computational Linguistics and Language Technology (GSCL)*, pages 115–124, Potsdam.

Shen, L., J. Xu, and R. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, OH.

Shieber, S. M. and Y. Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, pages 253–258, Helsinki.

Simard, M., N. Cancedda, B. Cavestro, M. Dymetman, E. Gaussier, C. Goutte, K. Yamada, P. Langlais, and A. Mauser. 2005. Translating with non-contiguous phrases. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 755–762, Vancouver.

Sixtus, A. and S. Ortmanns. 1999. High quality word graphs using forward-backward pruning. In *Proceedings of the IEEE International Conference Acoustics, Speech, and Signal Processing, 1999 - Volume 02*, pages 593–596, Phoenix, AZ.

Skut, W., B. Krenn, T. Brants, and H. Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 88–95, Washington, DC.

Smith, D. A. and J. Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 23–30, New York City.

Smith, D. A. and J. Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 822–831, Singapore.

Smith, N. A. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD.

Snyder, B., T. Naseem, and R. Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 73–81, Suntec.

Søgaard, A. and J. Kuhn. 2009. Empirical lower bounds on alignment error rates in syntax-based machine translation. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation (SSST-3) at NAACL HLT 2009*, pages 19–27, Boulder, CO.

Spitkovsky, V. I., H. Alshawi, A. X. Chang, and D. Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1,281–1,290, Edinburgh.

Spitkovsky, V. I., H. Alshawi, and D. Jurafsky. 2010. From baby steps to leapfrog: How "less is More" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, CA.

Stolcke, A. 2002. SRILM—An extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 901–904, Denver, CO.

Su, J., Y. Liu, H. Mi, H. Zhao, Y. Lü, and Q. Liu. 2010. Dependency-based

bracketing transduction grammar for statistical machine translation. In *Coling 2010: Posters*, pages 1,185–1,193, Beijing.

Tesnière, L. 1959. *Élément de Syntaxe Structurale*. Klincksieck.

Toutanova, K., D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 173–180, Edmonton.

Tromble, R., S. Kumar, F. Och, and W. Macherey. 2008. Lattice minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 620–629, Honolulu, HI.

Tsochantaridis, I., T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Tu, Z., Y. Liu, Y. Hwang, Q. Liu, and S. Lin. 2010. Dependency forest for statistical machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1,092–1,100, Beijing.

Ueffing, N., F. J. Och, and H. Ney. 2002. Generation of word graphs in statistical machine translation. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, pages 156–163, Philadelphia, PA.

Wang, M., N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague.

Weiss, D., B. Sapp, and B. Taskar. 2010. Sidestepping intractable inference with structured ensemble cascades. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2,415–2,423, Vancouver.

Wellington, B., S. Waxmonsky, and I. D. Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 977–984, Sydney.

Woodsend, K., Y. Feng, and M. Lapata. 2010. Title generation with quasi-synchronous grammar. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 513–523, Cambridge, MA.

Woodsend, K. and M. Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 409–420, Edinburgh.

Wu, D. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Wu, Y., Q. Zhang, X. Huang, and L. Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1,533–1,541, Singapore.

Xie, J., H. Mi, and Q. Liu. 2011. A novel dependency-to-string model for statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 216–226, Edinburgh.

Xiong, D., Q. Liu, and S. Lin. 2007. A dependency treelet string correspondence model for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 40–47, Prague.

Yadollahpour, P., D. Batra, and G. Shakhnarovich. 2013. Discriminative re-ranking of diverse segmentations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1,923–1,930, Portland, OR.

Yamada, H. and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206, Nancy, France.

Yamada, K. and K. Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530, Toulouse.

Yarowsky, D. and G. Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, pages 1–8, Pittsburgh, PA.

Yarowsky, D., G. Ngai, and R. Wicentoswki. 2001. Inducing multilingual text analysis

tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*, pages 1–8, San Diego, CA.

Zhang, J., F. Zhai, and C. Zong. 2011. Augmenting string-to-tree translation models with fuzzy use of source-side syntax. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 204–215, Edinburgh.

Zhang, Y. 2009. *Structured Language Models for Statistical Machine Translation*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

Zollmann, A. 2011. *Learning Multiple-Nonterminal Synchronous Grammars for Statistical Machine Translation*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

Zollmann, A. and A. Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 138–141, New York City.

Zollmann, A., A. Venugopal, F. J. Och, and J. Ponte. 2008. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1,145–1,152, Manchester.

Zollmann, A. and S. Vogel. 2011. A word-class approach to labeling PSCFG rules for machine translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1–11, Portland, OR.