Quasi-Synchronous Phrase Dependency Grammars for Machine Translation

Kevin Gimpel Noah A. Smith
Language Technologies Institute
Carnegie Mellon Univeristy
Pittsburgh, PA 15213, USA
{kgimpel,nasmith}@cs.cmu.edu

Abstract

We present a quasi-synchronous dependency grammar (Smith and Eisner, 2006) for machine translation in which the leaves of the tree are *phrases* rather than words as in previous work (Gimpel and Smith, 2009). This formulation allows us to combine structural components of phrase-based and syntax-based MT in a single model. We describe a method of extracting phrase dependencies from parallel text using a target-side dependency parser. For decoding, we describe a coarse-to-fine approach based on lattice dependency parsing of phrase lattices. We demonstrate performance improvements for Chinese-English and Urdu-English translation over a phrase-based baseline. We also investigate the use of *unsupervised* dependency parsers, reporting encouraging preliminary results.

1 Introduction

Two approaches currently dominate statistical machine translation (MT) research. Phrase-based models (Koehn et al., 2003) excel at capturing local reordering phenomena and memorizing multi-word translations. Models that employ syntax or syntax-like representations (Chiang, 2005; Galley et al., 2006; Zollmann and Venugopal, 2006; Huang et al., 2006) handle long-distance reordering better than phrase-based systems (Auli et al., 2009) but often require constraints on the formalism or rule extraction method in order to achieve computational tractability. As a result, certain instances of syntactic divergence are more naturally handled by phrase-based systems (DeNeefe et al., 2007).

In this paper we present a new way of combining the advantages of phrase-based and syntax-based MT. We propose a model in which phrases are organized into a tree structure inspired by dependency syntax. Instead of standard dependency trees in which *words* are vertices, our trees have *phrases* as vertices. We describe a simple heuristic to extract phrase dependencies from an aligned parallel corpus parsed on the target side, and use them to compute target-side tree features. We define additional string-to-tree features and, if a source-side dependency parser is available, tree-to-tree features to capture properties of how phrase dependencies interact with reordering.

To leverage standard phrase-based features alongside our novel features, we require a formalism that supports flexible feature combination and efficient decoding. Quasi-synchronous grammar (QG) provides this backbone (Smith and Eisner, 2006); we describe a coarse-to-fine approach for decoding within this framework, advancing substantially over earlier QG machine translation systems (Gimpel and Smith, 2009). The decoder involves generating a phrase lattice (Ueffing et al., 2002) in a coarse pass using a phrase-based model, followed by lattice dependency parsing of the phrase lattice. This approach allows us to feasibly explore the combined search space of segmentations, phrase alignments, and target phrase dependency trees.

Our experiments demonstrate an average improvement of +0.65 BLEU in Chinese-English translation across three test sets and an improvement of +0.75 BLEU in Urdu-English translation over a phrase-based baseline. We also describe experiments in which we replace supervised dependency parsers with *unsupervised* parsers, reporting promising results: using a supervised Chinese parser and a state-of-the-art unsupervised English parser provides our best results, giving an averaged gain of +0.79 BLEU over the baseline. We also discuss how our model improves translation quality and discuss future possibilities for combining approaches to machine translation using our framework.

2 Related Work

We previously applied quasi-synchronous grammar to machine translation (Gimpel and Smith, 2009), but that system performed translation fundamentally at the word level. Here we generalize that model to function on phrases, enabling a tighter coupling between the phrase segmentation and syntactic structures. We also present a decoder efficient enough to scale to large data sets and present performance improvements in large-scale experiments over a state-of-the-art phrase-based baseline.

Aside from QG, there have been many efforts to use dependency syntax in machine translation. Quirk et al. (2005) used a source-side dependency parser and projected automatic parses across word alignments in order to model dependency syntax on phrase pairs. Shen et al. (2008) presented an extension to Hiero (Chiang, 2005) in which rules have target-side dependency syntax and therefore enable the use of a dependency language model.

More recently, researchers have sought the benefits of dependency syntax while preserving the advantages of phrase-based models, such as efficiency and coverage. Galley and Manning (2009) loosened standard assumptions about dependency parsing so that the efficient left-to-right decoding procedure of phrase-based translation could be retained while a dependency language model is incorporated. Carreras and Collins (2009) presented a string-to-dependency system that permits non-projective dependency trees (thereby allowing a larger space of translations) and use a rule extraction procedure that includes rules for every phrase in the phrase table.

We take an additional step in this direction by working with dependency grammars on the phrases themselves, thereby bringing together the structural components of phrase-based and dependency-based MT in a single model. While others have worked on combining rules from multiple syntax-based systems (Liu et al., 2009) or using posteriors from multiple models to score translations (DeNero et al., 2010), we are not aware of any other work that seeks to directly integrate phrase-based and syntax-based machine translation at the modeling level.[1]

3 Model

Given a sentence $\boldsymbol{s}$ and its dependency tree $\tau_{\boldsymbol{s}}$, we formulate the translation problem as finding the target sentence $\boldsymbol{t}^*$, the segmentation $\boldsymbol{\gamma}^*$ of $\boldsymbol{s}$ into phrases, the segmentation $\boldsymbol{\phi}^*$ of $\boldsymbol{t}^*$ into phrases, the dependency tree $\tau^*_{\boldsymbol{\phi}}$ on the target phrases $\boldsymbol{\phi}^*$, and the one-to-one phrase alignment $\boldsymbol{a}^*$ such that

$$\langle \boldsymbol{t}^*, \boldsymbol{\gamma}^*, \boldsymbol{\phi}^*, \tau^*_{\boldsymbol{\phi}}, \boldsymbol{a}^* \rangle = \operatorname*{argmax}_{\langle \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \rangle} p(\boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}})$$

We use a linear model (Och and Ney, 2002):

$$p(\boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}}) \propto \exp\{\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}, \tau_{\boldsymbol{s}}, \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a})\}$$

where $\boldsymbol{g}$ is a vector of arbitrary feature functions on the full set of structures and $\boldsymbol{\theta}$ holds corresponding feature weights. Table 1 summarizes our notation.

In modeling $p(\boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}})$, we make use of **quasi-synchronous grammar** (QG; Smith and Eisner, 2006). Given a source sentence and its parse, a QG induces a probabilistic monolingual grammar over sentences "inspired" by the source sentence and tree. We denote this grammar by $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$; its (weighted) language is the set of translations of $\boldsymbol{s}$.

Quasi-synchronous grammar makes no restrictions on the form of the target monolingual grammar, though dependency grammars have been used in most previous applications of QG (Wang et al., 2007; Das and Smith, 2009; Smith and Eisner, 2009), including previous work in MT (Smith and Eisner, 2006; Gimpel and Smith, 2009). We previously presented a word-based machine translation model based on a quasi-synchronous dependency grammar. However, it is well-known in the MT community that translation quality is improved when larger units are modeled. Therefore, we use a dependency grammar in which the leaves are *phrases* rather than words.

We define a **phrase dependency grammar** as a model $p(\boldsymbol{\phi}, \tau_{\boldsymbol{\phi}} | \boldsymbol{t})$ over the joint space of segmentations of a sentence into phrases and dependency trees on the phrases.[2] Phrase dependency grammars

[1]Dymetman and Cancedda (2010) present a formal analysis of the problem of intersecting phrase-based and hierarchical translation models, but do not provide experimental results.

[2]We restrict our attention to projective trees in this paper, but the generalization to non-projective trees is easily made.

$\boldsymbol{s} = \langle s_1, \ldots, s_n \rangle$	source language sentence
$\boldsymbol{t} = \langle t_1, \ldots, t_m \rangle$	target language sentence, translation of $\boldsymbol{s}$
$\boldsymbol{\gamma} = \langle \gamma_1, \ldots, \gamma_{n'} \rangle$ $\forall i, \gamma_i = \langle s_j, \ldots, s_k \rangle$ s.t. $\gamma_1 \cdot \ldots \cdot \gamma_{n'} = \boldsymbol{s}$	segmentation of $\boldsymbol{s}$ into phrases
$\boldsymbol{\phi} = \langle \phi_1, \ldots, \phi_{m'} \rangle$ $\forall i, \phi_i = \langle t_j, \ldots, t_k \rangle$ s.t. $\phi_1 \cdot \ldots \cdot \phi_{m'} = \boldsymbol{t}$	segmentation of $\boldsymbol{t}$ into phrases
$\tau_{\boldsymbol{s}} : \{1, \ldots, n\} \rightarrow \{0, \ldots, n\}$	dependency tree on source words $\boldsymbol{s}$, where $\tau_{\boldsymbol{s}}(i)$ is the index of the parent of word s_i (0 is the root, \$)
$\tau_{\boldsymbol{\phi}} : \{1, \ldots, m'\} \rightarrow \{0, \ldots, m'\}$	dependency tree on target phrases $\boldsymbol{\phi}$, where $\tau_{\boldsymbol{\phi}}(i)$ is the index of the parent of phrase ϕ_i
$\boldsymbol{a} : \{1, \ldots, m'\} \rightarrow \{1, \ldots, n'\}$	one-to-one alignment from phrases in $\boldsymbol{\phi}$ to phrases in $\boldsymbol{\gamma}$
$\boldsymbol{\theta} = \langle \boldsymbol{\lambda}, \boldsymbol{\psi} \rangle$	parameters of the full model ($\boldsymbol{\lambda}$ = phrase-based, $\boldsymbol{\psi}$ = QPDG)

Table 1: Key notation.

have recently been used by Wu et al. (2009) for feature extraction for opinion mining. When used for translation modeling, they allow us to capture phenomena like local reordering and idiomatic translations within each phrase as well as long-distance relationships among the phrases in a sentence.

We then define a **quasi-synchronous phrase dependency grammar** (QPDG) as a conditional model $p(\boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}})$ that induces a probabilistic monolingual phrase dependency grammar over sentences inspired by the source sentence and (lexical) dependency tree. The source and target sentences are segmented into phrases and the phrases are aligned in a one-to-one alignment.

We note that we actually depart here slightly from the original definition of QG. The alignment variable in QG links target tree nodes to source tree nodes. However, we never commit to a source phrase dependency tree, instead using a source lexical dependency tree output by a dependency parser, so our alignment variable $\boldsymbol{a}$ is a function from target tree nodes (phrases in $\boldsymbol{\phi}$) to source phrases in $\boldsymbol{\gamma}$, which might not be source tree nodes. The features in our model may consider a large number of source phrase dependency trees as long as they are consistent with $\tau_{\boldsymbol{s}}$.

4 Features

Our model contains all of the standard phrase-based features found in systems like Moses (Koehn et al., 2007), including 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.

We now describe in detail the additional features

\$ ← said :	\$ ← we should
\$ ← said that	\$ ← has been
\$ ← is a	- us → relations
\$ ← will be	\$ ← he said
\$ ← it is	cross - strait → relations
\$ ← this is	\$ ← pointed out that
\$ ← we must	, and → is
the → united states	the chinese → government
the → development of	\$ ← is the
the two → countries	\$ ← said ,
he → said :	one - china → principle
\$ ← he said :	sino - us → relations

Table 2: Most frequent phrase dependencies with at least 2 words in one of the phrases (dependencies in which one phrase is entirely punctuation are not shown). \$ indicates the root of the tree.

in our model that are used to score phrase dependency trees. We shall refer to these as QPDG features and will find it useful later to notationally distinguish their feature weights from those of the phrase-based model. We use $\boldsymbol{\lambda}$ for weights of the standard phrase-based model features and $\boldsymbol{\psi}$ for weights of the QPDG features. We include three categories of features, differentiated by what pieces of structure they consider.

4.1 Target Tree Features

We first include features that only consider $\boldsymbol{t}$, $\boldsymbol{\phi}$, and $\tau_{\boldsymbol{\phi}}$. These features can be categorized as "syntactic language model" features (Shen et al., 2008; Galley and Manning, 2009), though unlike previous work our features model both the phrase segmentation and dependency structure. Typically, these sorts of features are probabilities estimated from a corpus parsed using a supervised parser. However, there do not currently exist treebanks with annotated phrase

, $\rightarrow$ made up	0.057
he $\rightarrow$ made up	0.021
supreme court $\rightarrow$ made up	0.014
court $\rightarrow$ made up	0.014
in september 2000 $\rightarrow$ made up	0.014
in september 2000 , $\rightarrow$ made up	0.014
made up $\leftarrow$ of	0.065
made up $\leftarrow$.	0.029
made up $\leftarrow$,	0.016
made up $\leftarrow$ mind to	0.01

Table 3: Most probable child phrases for the parent phrase "made up" for each direction, sorted by the conditional probability of the child phrase given the parent phrase and direction.

dependency trees.

Our solution is to use a standard supervised dependency parser and extract phrase dependencies using bilingual information.[3] We begin by obtaining symmetrized word alignments and extracting phrase pairs using the standard heuristic from phrase-based MT (Koehn et al., 2003). Given the set of extracted phrase pairs for a sentence, denote by W the set of unique target-side phrases among them. We parse the target sentence with a dependency parser and, for each pair of phrases $u, v \in W$, we extract a phrase dependency (along with its direction) if u and v do not overlap and there is at least one lexical dependency between a word in u and a word in v. If there are lexical dependencies in both directions, we extract a phrase dependency only for the single longest one. Since we use a projective dependency parser, the longest lexical dependency between two phrases is guaranteed to be unique. Table 2 shows a listing of the most frequent phrase dependencies extracted (lexical dependencies are omitted).

We note that during training we never explicitly commit to any single phrase dependency tree for a target sentence. Rather, we extract phrase dependencies from all phrase dependency trees consistent with the word alignments and the lexical dependency tree. Thus we treat phrase dependency trees analogously to phrase segmentations in standard phrase extraction.

We perform this procedure on all sentence pairs in the parallel corpus. Given a set of extracted phrase dependencies of the form $\langle u, v, d \rangle$, where u is the head phrase, v is the child phrase, and $d \in \{\mathit{left}, \mathit{right}\}$ is the direction, we then estimate conditional probabilities $p(v|u, d)$ using relative frequency estimation. Table 3 shows the most probable child phrases for an example parent phrase. To combat data sparseness, we perform the same procedure with each word replaced by its word cluster ID obtained from Brown clustering (Brown et al., 1992).

We include a feature in the model for the sum of the scaled log-probabilities of each attachment:

$$\sum_{i=1}^{m'} \max\left(0, C + \log p(\phi_i | \phi_{\tau_\phi(i)}, d(i)\right) \qquad (1)$$

where $d(i) = I[\tau_\phi(i) - i > 0]$ is the direction of the dependency arc.

Although we use log-probabilities in this feature function, we first add a constant C to each to ensure they are all positive.[4] The $\max$ expression protects unseen parent-child phrase dependencies from causing the score to be negative infinity. Our motivation is a desire for the features to be used to prefer one derivation over another but not to rule out a derivation completely if it merely happens to contain a dependency unobserved in the training data.

We also include lexical weighting features similar to those used in phrase-based MT (Koehn et al., 2003). Whenever we extract a phrase dependency, we extract the longest lexical dependency contained within it. For all $\langle$parent, child, direction$\rangle$ lexical dependency tuples $\langle x, y, d \rangle$, we estimate conditional probabilities $p_{lex}(y|x, d)$ from the parsed corpus using relative frequency estimation. Then, for a phrase dependency with longest lexical dependency $\langle x, y, d \rangle$, we add a feature for $p_{lex}(y|x, d)$ to the model, using a formula similar to Eq. 1. Different instances of a phrase dependency may have different lexical dependencies extracted with them. We add the lexical weight for the most frequent, breaking ties by choosing the lexical dependency that maximizes $p(y|x, d)$, as was also done by Koehn et al. (2003).

In all, we include 4 target tree features: one for phrase dependencies, one for lexical dependencies,

[3]For a monolingual task, Wu et al. (2009) used a shallow parser to convert lexical dependencies from a dependency parser into phrase dependencies.

[4]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.

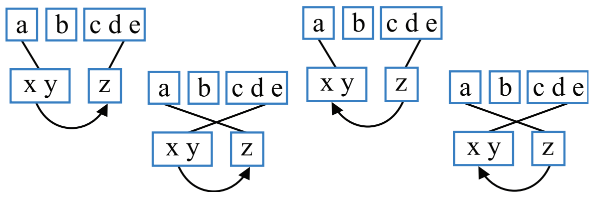


Figure 1: String-to-tree configurations; each is associated with a feature that counts its occurrences in a derivation.

```
Input: sentence $\boldsymbol{s}$, dependency parse $\tau_{\boldsymbol{s}}$, coarse
        parameters $\boldsymbol{\lambda}_M$, fine parameters $\langle \boldsymbol{\lambda}, \boldsymbol{\psi} \rangle$
Output: translation $\boldsymbol{t}$
$L_{\text{MERT}} \leftarrow$ GenerateLattices $(\boldsymbol{s}, \boldsymbol{\lambda}_M)$;
$L_{\text{FB}} \leftarrow$ FBPrune $(L_{\text{MERT}}, \boldsymbol{\lambda}_M)$;
$\langle \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \rangle \leftarrow$ QGDEPPARSE$(L_{\text{FB}}, \langle \boldsymbol{\lambda}, \boldsymbol{\psi} \rangle)$;
return $\boldsymbol{t}$;
```
Algorithm 1: CoarseToFineDecode

and the same features computed from a transformed version of the corpus in which each word is replaced by its Brown cluster.

4.2 String-to-Tree Configurations

We consider features that count instances of reordering configurations involving phrase dependencies. In addition to the target-side structures, these features consider $\boldsymbol{\gamma}$ and $\boldsymbol{a}$, though not $\boldsymbol{s}$ or $\tau_{\boldsymbol{s}}$. For example, when building a parent-child phrase dependency with the child to the left, one feature value is incremented if their aligned source-side phrases are in the same order. This configuration is the leftmost in Fig. 1; we include features for the other three configurations there as well, for a total of 4 features in this category.

4.3 Tree-to-Tree Configurations

We include features that consider $\boldsymbol{s}$, $\boldsymbol{\gamma}$, and $\tau_{\boldsymbol{s}}$ in addition to $\boldsymbol{t}$, $\boldsymbol{\phi}$, and $\tau_{\boldsymbol{\phi}}$. We begin with features for each of the quasi-synchronous configurations from Smith and Eisner (2006), adapted to phrase dependency grammars. That is, for a parent-child pair $\langle \tau_{\boldsymbol{\phi}}(i), i \rangle$ in $\tau_{\boldsymbol{\phi}}$, we consider the relationship between $\boldsymbol{a}(\tau_{\boldsymbol{\phi}}(i))$ and $\boldsymbol{a}(i)$, the source-side phrases to which $\tau_{\boldsymbol{\phi}}(i)$ and i align. We use the following named configurations from Smith and Eisner: root-root, parent-child, child-parent, grandparent-grandchild, sibling, and c-command.[5] We define a feature to count instances of each of these configurations, including an additional feature for "other" configurations that do not fit into these categories.[6]

When using a QPDG, there are multiple ways to compute tree-to-tree configuration features, since we use a phrase dependency tree for the target side, a lexical dependency tree for the source side, and a phrase alignment. We use the following heuristic approach. Given a pair of source words, one with index j in source phrase $\boldsymbol{a}(\tau_{\boldsymbol{\phi}}(i))$ and the other with index k in source phrase $\boldsymbol{a}(i)$, we have a parent-child configuration if $\tau_{\boldsymbol{s}}(k) = j$; if $\tau_{\boldsymbol{s}}(j) = k$, 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 and other c-command configurations, the shared parent or ancestor must also be outside both phrases.

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 phrase dependency attachment.

Finally, we include features that consider the dependency path distance between phrases in the source-side dependency tree that are aligned to parent-child pairs in $\tau_{\boldsymbol{\phi}}$. We include a feature that sums, for each target phrase i, the inverse of the minimum undirected path length between each word in $\boldsymbol{a}(i)$ and each word in $\tau_{\boldsymbol{\phi}}(\boldsymbol{a}(i))$. The minimum undirected path length is defined as the number of dependency arcs that must be crossed to travel from one word to the other in $\tau_{\boldsymbol{s}}$. We use one feature for undirected path length and one other for directed path length. If there is no (un)directed path from a word in $\boldsymbol{a}(i)$ to a word in $\tau_{\boldsymbol{\phi}}(\boldsymbol{a}(i))$, we use ∞ as the minimum length.

There are 15 features in this category, for a total of 23 QPDG features.

[5] See Fig. 3 in Smith and Eisner (2006) for illustrations.

[6] We actually include two versions of each configuration feature other than "root-root": one for the source phrases being in the same order as the target phrases and one for them being swapped.

5 Decoding

For a QPDG model, decoding consists of finding the highest-scoring tuple $\langle \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \rangle$ for an input sentence $\boldsymbol{s}$ and its parse $\tau_{\boldsymbol{s}}$, i.e., finding the most probable derivation under the $\boldsymbol{s}/\tau_{\boldsymbol{s}}$-specific grammar $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$. We follow Gimpel and Smith (2009) in constructing a lattice to represent $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$ and using lattice parsing to search for the best derivation, but we construct the lattice differently and employ a coarse-to-fine strategy (Petrov, 2009) to speed up decoding.

It has become common in recent years for MT researchers to exploit efficient data structures for encoding concise representations of the pruned search space of the model, such as phrase lattices for phrase-based MT (Ueffing et al., 2002; Macherey et al., 2008; Tromble et al., 2008). Each edge in a phrase lattice corresponds to a phrase pair and each path through the lattice corresponds to a tuple $\langle \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \boldsymbol{a} \rangle$ for the input $\boldsymbol{s}$. Decoding for a phrase lattice consists of finding the highest-scoring path, which is done using dynamic programming. To also maximize over $\tau_{\boldsymbol{\phi}}$, we perform lattice dependency parsing, which allows us to search over the space of tuples $\langle \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \boldsymbol{a}, \tau_{\boldsymbol{\phi}} \rangle$. Given the lattice and $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$, lattice parsing is a straightforward generalization of the standard arc-factored dynamic programming algorithm from Eisner (1996).

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.[7] 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 approximate search based on coarse-to-fine decoding. We now discuss each step of this procedure; an outline is shown as Alg. 1.

Pass 1: Lattice Pruning After generating phrase lattices using a phrase-based MT system, we prune lattice edges using forward-backward pruning (Sixtus and Ortmanns, 1999), which has also been used in previous work using phrase lattices (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. Max-marginals offer the advantage that the best path in the lattice is preserved during pruning. For each lattice, we use a grid search to find the most liberal threshold that leaves fewer than 1000 edges in the resulting lattice. As complexity is quadratic in E, forcing E to be less than 1000 improves runtime substantially. After pruning, the lattices still contain more than 10^{16} paths on average and oracle BLEU scores are typically 12-15 points higher than the model-best paths.

Pass 2: Parent Ranking Given a pruned lattice, we then remove some candidate dependency arcs from consideration. It is common in dependency parsing to use a coarse model to rank the top k parents for each word, and to only consider these during parsing (Martins et al., 2009; Bergsma and Cherry, 2010). Unlike string parsing, our phrase lattices impose several types of constraints on allowable 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. To handle constraints like these, we first use the Floyd-Warshall algorithm (Floyd, 1962) to find the best score between every pair of nodes in the lattice. This algorithm also tells us whether each edge is reachable from each other edge, allowing us to immediately prune dependency arcs between edges that are unreachable from each other.

After eliminating impossible arcs, we turn to pruning away unlikely ones. In standard (string) dependency parsing, every word is assigned a parent. In lattice parsing, however, most lattice edges will not be assigned any parent. Certain lattice edges are much more likely to be contained within paths, so we allow some edges to have more candidate parent edges than others. We introduce hyperparameters α, β, and μ to denote, respectively, the minimum, maximum, and average number of parent edges to be considered for each lattice edge ($\alpha \leq \mu \leq \beta$). We rank the full set of E^2 arcs according to their scores (using the QPDG features and their weights $\boldsymbol{\psi}$) and choose the top μE of these arcs while ensuring that each edge has at least α and at most β potential parent edges.

This step reduces the time complexity from $O(E^2V)$ to $O(\mu EV)$, where $\mu < E$. In our experiments, we set $\mu = 300$, $\alpha = 100$, and $\beta = 400$.

[7]To prevent confusion, we use the term *edge* to refer to a phrase lattice edge and *arc* to refer to a parent-child dependency in the phrase dependency tree.

```
**Input**: tuning set $D = \langle S, T \rangle$, initial weights $\boldsymbol{\lambda}_0$ for
        coarse model, initial weights $\boldsymbol{\psi}_0$ for
        additional features in fine model
**Output**: coarse model learned weights: $\boldsymbol{\lambda}_M$, fine
        model learned weights: $\langle \boldsymbol{\lambda}^*, \boldsymbol{\psi}^* \rangle$
$\boldsymbol{\lambda}_M \leftarrow$ MERT ($S$, $T$, $\boldsymbol{\lambda}_0$, 100, MOSES);
$L_{\text{MERT}} \leftarrow$ GenerateLattices ($S$, $\boldsymbol{\lambda}_M$);
$L_{\text{FB}} \leftarrow$ FBPrune ($L_{\text{MERT}}$, $\boldsymbol{\lambda}_M$);
$\langle \boldsymbol{\lambda}^*, \boldsymbol{\psi}^* \rangle \leftarrow$
    MERT ($L_{\text{FB}}$, $T$, $\langle \boldsymbol{\lambda}_M, \boldsymbol{\psi}_0 \rangle$, 200, QGDEPPARSE);
**return** $\boldsymbol{\lambda}_M$, $\langle \boldsymbol{\lambda}^*, \boldsymbol{\psi}^* \rangle$;
```

Algorithm 2: CoarseToFineTrain

Pass 3: Lattice Dependency Parsing After completing the coarse passes, we parse using bottom-up dynamic programming based on the agenda algorithm (Nederhof, 2003; Eisner et al., 2005). We only consider arcs that survived the filtering in Pass 2. We weight agenda items by the sum of their scores and the Floyd-Warshall best path scores both from the start node of the lattice to the beginning of the item and the end of the item to any final node. This heuristic helps us to favor exploration of items that are highly likely under the phrase-based model.

If the score of the partial structure can only get worse when combining it with other structures (e.g., in a PCFG), then the first time that we pop an item of type GOAL from the agenda, we are guaranteed to have the best parse. However, in our model, some features are positive and others negative, making this property no longer hold; as a result, GOAL items may be popped out of order from the agenda. Therefore, we use an approximation, simply popping G GOAL items from the agenda and then stopping. The items are sorted by their scores and the best is returned by the decoder (or the k best in the case of MERT). In our experiments, we set $G = 4000$.

The combined strategy yields average decoding times in the range of 30 seconds per sentence, which is comparable to other syntax-based MT systems.

6 Training

For tuning the coarse and fine parameters, we use minimum error rate training (MERT; Och, 2003) in a procedure shown as Alg. 2. We first use MERT to train parameters for the coarse phrase-based model used to generate phrase lattices. Then, after generating the lattices, we prune them and run MERT a second time to tune parameters of the fine model, which includes all phrase-based and QPDG parameters. The arguments to MERT are a vector of source sentences (or lattices), a vector of target sentences, the initial parameter values, the size of the k-best list, and finally the decoder. We initialize $\boldsymbol{\lambda}$ to the default Moses feature weights and for $\boldsymbol{\psi}$ we initialize the two target phrase dependency weights to 0.004, the two lexical dependency weights to 0.001, and the weights for all configuration features to 0.0. Our training procedure requires two executions of MERT, and the second typically takes more iterations to converge (10 to 20 is typical) than the first due to the use of a larger feature set and increased possibility for search error due to the enlarged search space.

7 Experiments

For experimental evaluation, we consider Chinese-to-English (ZH-EN) and Urdu-to-English (UR-EN) translation and compare our system to Moses (Koehn et al., 2007). For ZH-EN, we used 303k sentence pairs from the FBIS corpus (LDC2003E14). We segmented the Chinese data using the Stanford Chinese segmenter in "CTB" mode (Chang et al., 2008), giving us 7.9M Chinese words and 9.4M English words. For UR-EN, we used parallel data from the NIST MT08 evaluation consisting of 1.2M Urdu words and 1.1M English words.

We trained a baseline Moses system using default settings and features. Word alignment was performed using GIZA++ (Och and Ney, 2003) in both directions and the `grow-diag-final-and` heuristic was used to symmetrize the alignments. We used a max phrase length of 7 when extracting phrases. Trigram language models were estimated using the SRI language modeling toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1998). To estimate language models for each language pair, we used the English side of the parallel corpus concatenated with 200M words of randomly-selected sentences from the Gigaword v4 corpus (excluding the NY Times and LA Times).

We used this baseline Moses system to generate phrase lattices for our system, so our model includes all of the Moses features in addition to the

	MT03 (tune)	MT02	MT05	MT06	Average
Moses	33.84	33.35	31.81	28.82	31.33
QPDG (TT)	34.63 (+0.79)	34.10 (+0.75)	32.15 (+0.34)	29.33 (+0.51)	31.86 (+0.53)
QPDG (TT+S2T+T2T)	34.98 (+1.14)	34.26 (+0.91)	32.34 (+0.53)	29.35 (+0.53)	31.98 (+0.65)

Table 4: Chinese-English Results (% BLEU).

QPDG features described in §4. In our experiments, we compare our QPDG system (lattice parsing on each lattice) to the Moses baseline (finding the best path through each lattice). The conventional wisdom holds that hierarchical phrase-based translation (Chiang, 2005) performs better than phrase-based translation for language pairs that require large amounts of reordering, such as ZH-EN and UR-EN. However, researchers have shown that this performance gap diminishes when using a larger distortion limit (Zollmann et al., 2008) and may disappear entirely when using a lexicalized reordering model (Lopez, 2008; Galley and Manning, 2010). So, we increase the Moses distortion limit from 6 (the default) to 10 and use Moses' default lexicalized reordering model (Koehn et al., 2005).

We parsed the Chinese text using the Stanford parser (Levy and Manning, 2003) and the English text using TurboParser (Martins et al., 2009). We note that computing our features requires parsing the target (English) side of the parallel text, but not the source side. We only need to parse the source side of the tuning and test sets, and the only features that look at the source-side parse are those from §4.3.

To obtain Brown clusters for the target tree features in §4.1, we used code from Liang (2005).[8] We induced 100 clusters from the English side of the parallel corpus concatenated with 10M words of randomly-selected Gigaword sentences. Only words that appeared at least twice in this data were considered during clustering. An additional cluster was created for all other words; this allowed us to use phrase dependency cluster features even for out-of-vocabulary words. We used a max phrase length of 7 when extracting phrase dependencies to match the max phrase length used in phrase extraction. Approximately 87M unique phrase dependencies were extracted from the ZH-EN data and 7M from the UR-EN data.

We tuned the weights of our model using the pro-

[8] `http://www.cs.berkeley.edu/~pliang/software`

	Dev (tune)	MT09
Moses	24.21	23.56
QPDG (TT+S2T)	24.94 (+0.73)	24.31 (+0.75)

Table 5: Urdu-English Results (% BLEU).

cedure described in §6. For ZH-EN we used MT03 for tuning and MT02, MT05, and MT06 for testing. For UR-EN we used half of the documents (882 sentence pairs) from the MT08 test set for tuning ("Dev") and MT09 for testing. We evaluated translation output using case-insensitive IBM BLEU (Papineni et al., 2001).

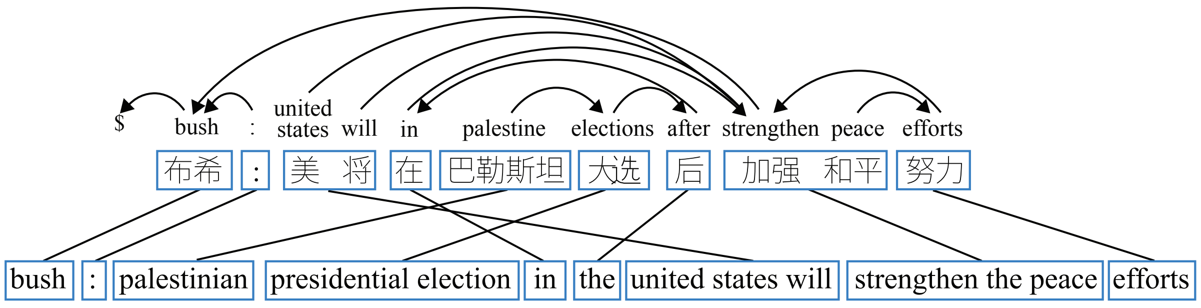
7.1 Results

Results for ZH-EN and UR-EN translation are shown in Tables 4 and 5. We show results when using only the target tree features from §4.1 (TT), as well as when adding the string-to-tree features from §4.2 (S2T) and the tree-to-tree features from §4.3 (T2T). We note that T2T features are unavailable for UR-EN because we do not have an Urdu parser. We find that we can achieve moderate but consistent improvements over the baseline Moses system, for an average increase of 0.65 BLEU points for ZH-EN and 0.75 for UR-EN.

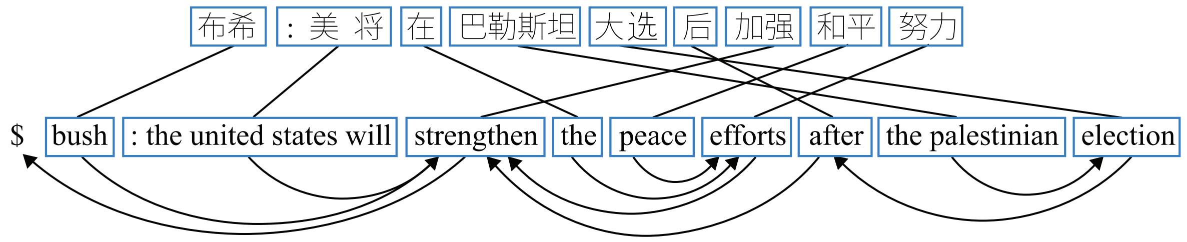
Fig. 2 shows an example sentence from the MT05 test set along with its translation output and derivations produced by Moses and our QPDG system with the full feature set. This example shows the kind of improvements that our system makes. In Chinese, modifiers such as prepositional phrases and clauses are generally placed in front of the words they modify, frequently the opposite of English. In addition, Chinese occasionally uses postpositions where English uses prepositions. The Chinese sentence in Fig. 2 exhibits both of these, as the prepositional phrase "after the Palestinian election" appears before the verb "strengthen" in the Chinese sentence and "after" appears as a postposition. Moses (Fig. 2(a)) does not properly reorder the prepositional phrase, while our system (Fig. 2(b)) properly handles both reorderings.[9] We shall discuss these

[9] Our system's derivation is not perfect, in that "in" is incor-

(a)



(b)



(c)	References	bush : us set to boost peace efforts after palestinian election
		bush : us to step up peace efforts after palestinian elections
		bush : u.s. will enhance peace efforts after palestinian election
		us to boost peace efforts after palestinian elections : bush

Figure 2: (a) Moses translation output along with γ, ϕ, and $\boldsymbol{a}$. An English gloss is shown above the Chinese sentence and above the gloss is shown the dependency parse from the Stanford parser. (b) QPDG system output with additional structure τ_ϕ. (c) reference translations.

types of improvements further in §8.

7.2 Unsupervised Parsing

Our results thus far use supervised parsers for both Chinese and English, but parsers are only available for a small fraction of the languages we would like to translate. Fortunately, unsupervised dependency grammar induction has improved substantially in recent years due to a flurry of recent research. While attachment accuracies on standard treebank test sets are still relatively low, it may be the case that even though unsupervised parsers do not match treebank annotations very well, they may perform well when used for extrinsic applications. We believe that syntax-based MT offers a compelling platform for development and extrinsic evaluation of unsupervised parsers.

In this paper, we use the standard dependency model with valence (DMV; Klein and Manning, 2004). When training is initialized using the output of a simpler, concave dependency model, the DMV can approach state-of-the-art unsupervised accuracy (Gimpel and Smith, 2011). For English, the resulting parser achieves 53.1% attachment accuracy on Section 23 of the Penn Treebank (Marcus et al., 1993), which approaches the 55.7% accuracy of a recent state-of-the-art unsupervised model (Blunsom and Cohn, 2010). The Chinese parser, initialized and trained the same way, achieves 44.4%, which is the highest reported accuracy on the Chinese Treebank (Xue et al., 2004) test set.

Most unsupervised grammar induction models assume gold standard POS tags and sentences stripped of punctuation. We use the Stanford tagger (Toutanova et al., 2003) to obtain tags for both English and Chinese, parse the sentences without punctuation using the DMV, and then attach punctuation tokens to the root word of the tree in a post-processing step. For English, the predicted parents agreed with those of TurboParser for 48.7% of the tokens in the corpus.

We considered all four scenarios: supervised and unsupervised English parsing paired with supervised and unsupervised Chinese parsing. Table 6 shows

rectly translated and reordered, but the system was nonetheless able to use it to improve the fluency of the output.

		EN	
		unsupervised	supervised
ZH	unsupervised	31.18 (33.76)	31.86 (34.78)
	supervised	**32.12** (34.74)	31.98 (34.98)
Moses		31.33 (33.84)	

Table 6: Results when using unsupervised dependency parsers. Cells contain averaged % BLEU on the three test sets and % BLEU on tuning data (MT03) in parentheses.

Feature	Initial	Learned
Left child, same order	9.0	8.9
Left child, swap phrases	1.1	0.0
Right child, same order	7.3	7.3
Right child, swap phrases	1.6	2.3
Root-root	0.4	0.8
Parent-child	4.2	6.1
Child-parent	1.2	0.4
Grandparent-grandchild	1.0	0.2
Sibling	2.4	1.9
C-command	6.1	6.7
Other	1.5	0.9

Table 7: Average feature values across best translations of sentences in the MT03 tuning set, both before MERT (column 2) and after (column 3). "Same" versions of tree-to-tree configuration features are shown; the rarer "swap" features showed a similar trend.

BLEU scores averaged over the three test sets with tuning data BLEU in parentheses. Surprisingly, we achieve our best results when using the unsupervised English parser in place of the supervised one (+0.79 over Moses), while keeping the Chinese parser supervised. Competitive performance is also found by using the unsupervised Chinese parser and supervised English parser (+0.53 over Moses).

However, when using unsupervised parsers for both languages, performance was below that of Moses. During tuning for this configuration, we found that MERT struggled to find good parameter estimates, typically converging to suboptimal solutions after a small number of iterations. We believe this is due to the large number of features (37), the noise in the parse trees, and known instabilities of MERT. In future work we plan to experiment with training algorithms that are more stable and that can handle larger numbers of features.

8 Analysis

To understand what our model learns during MER training, we computed the feature vectors of the best derivation for each sentence in the tuning data at both the start and end of tuning. Table 7 shows these feature values averaged across all tuning sentences. The first four features are the configurations from Fig. 1, in order from left to right. From these rows, we can observe that the model learns to encourage swapping when generating right children and penalize swapping for left children. In addition to objects, right children in English are often prepositional phrases, relative clauses, or other modifiers; as we noted above, Chinese generally places these modifiers before their heads, requiring reordering during translation. Here the model appears to be learning this reordering behavior.

From the second set of features, we see that the model learns to favor producing dependency trees that are mostly isomorphic to the source tree, by favoring root-root and parent-child configurations at the expense of most others.

9 Discussion

In looking at BLEU score differences between the two systems, the unigram precisions were typically equal or only slightly different, while precisions for higher-order n-grams contained the bulk of the improvement. This suggests that our system is not finding substantially better translations for individual words in the input, but rather is focused on reordering the existing translations. This is not surprising given our choice of features, which focus on syntactic language modeling and syntax-based reordering. The obvious next step for our framework is to include bilingual rules that include source syntax (Quirk et al., 2005), target syntax (Shen et al., 2008), and syntax on both sides. Our framework allows integrating together all of these and other types of structures, with the ultimate goal of combining the strengths of multiple approaches to translation in a single model.

Acknowledgments

We thank Chris Dyer and the anonymous reviewers for helpful comments that improved this paper. This research was supported in part by the NSF 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

M. Auli, A. Lopez, H. Hoang, and P. Koehn. 2009. A systematic analysis of translation model search spaces. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*.

S. Bergsma and C. Cherry. 2010. Fast and accurate arc filtering for dependency parsing. In *Proc. of COLING*.

P. Blunsom and T. Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proc. of EMNLP*.

P. F. Brown, 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.

X. Carreras and M. Collins. 2009. Non-projective parsing for statistical machine translation. In *Proc. of EMNLP*.

P. Chang, M. Galley, and C. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proc. of the Third Workshop on Statistical Machine Translation*.

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

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL*.

D. Das and N. A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proc. of ACL-IJCNLP*.

S. DeNeefe, K. Knight, W. Wang, and D. Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proc. of EMNLP-CoNLL*.

J. DeNero, S. Kumar, C. Chelba, and F. J. Och. 2010. Model combination for machine translation. In *Proc. of NAACL*.

M. Dymetman and N. Cancedda. 2010. Intersecting hierarchical and phrase-based models of translation. formal aspects and algorithms. In *Proc. of SSST-4*.

J. Eisner, E. Goldlust, and N. A. Smith. 2005. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In *Proc. of HLT-EMNLP*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*.

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

M. Galley and C. D. Manning. 2009. Quadratic-time dependency parsing for machine translation. In *Proc. of ACL-IJCNLP*.

M. Galley and C. D. Manning. 2010. Accurate non-hierarchical phrase-based translation. In *Proc. of NAACL*.

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of COLING-ACL*.

K. Gimpel and N. A. Smith. 2009. Feature-rich translation by quasi-synchronous lattice parsing. In *Proc. of EMNLP*.

K. Gimpel and N. A. Smith. 2011. Concavity and initialization for unsupervised dependency grammar induction. Technical report, Carnegie Mellon University.

L. Huang, K. Knight, and A. Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*.

P. Koehn, A. Axelrod, A. Birch Mayne, C. Callison-Burch, M. Osborne, and D. Talbot. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *Proc. of IWSLT*.

P. Koehn, 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 *Proc. of ACL (demo session)*.

R. Levy and C. D. Manning. 2003. Is it harder to parse chinese, or the chinese treebank? In *Proc. of ACL*.

P. Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Y. Liu, H. Mi, Y. Feng, and Q. Liu. 2009. Joint decoding with multiple translation models. In *Proc. of ACL-IJCNLP*.

A. Lopez. 2008. Tera-scale translation models via pattern matching. In *Proc. of COLING*.

W. Macherey, F. Och, I. Thayer, and J. Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *EMNLP*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

A. F. T. Martins, N. A. Smith, and E. P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL*.

M.-J. Nederhof. 2003. Weighted deductive parsing and knuth's algorithm. *Computational Linguistics*, 29(1).

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL*.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1).

F. J. Och. 2003. Minimum error rate training for statistical machine translation. In *Proc. of ACL*.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

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

C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL*.

L. Shen, J. Xu, and R. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL*.

A. Sixtus and S. Ortmanns. 1999. High quality word graphs using forward-backward pruning. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*.

D. A. Smith and J. Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proc. of HLT-NAACL Workshop on Statistical Machine Translation*.

D. A. Smith and J. Eisner. 2009. Parser adaptation and projection with quasi-synchronous features. In *Proc. of EMNLP*.

A. Stolcke. 2002. SRILM—an extensible language modeling toolkit. In *Proc. of ICSLP*.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of HLT-NAACL*.

R. Tromble, S. Kumar, F. Och, and W. Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *EMNLP*.

N. Ueffing, F. J. Och, and H. Ney. 2002. Generation of word graphs in statistical machine translation. In *Proc. of EMNLP*.

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*.

Y. Wu, Q. Zhang, X. Huang, and L. Wu. 2009. Phrase dependency parsing for opinion mining. In *Proc. of EMNLP*.

N. Xue, F. Xia, F.-D. Chiou, and M. Palmer. 2004. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 10(4):1–30.

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

A. Zollmann, A. Venugopal, F. J. Och, and J. Ponte. 2008. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proc. of COLING*.