Feature-Rich Translation by Quasi-Synchronous Lattice Parsing

Kevin Gimpel and **Noah A. Smith**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{kgimpel,nasmith}@cs.cmu.edu

Abstract

We present a machine translation framework that can incorporate arbitrary features of both input and output sentences. The core of the approach is a novel decoder based on lattice parsing with quasi-synchronous grammar (Smith and Eisner, 2006), a syntactic formalism that does not require source and target trees to be isomorphic. Using generic approximate dynamic programming techniques, this decoder can handle "non-local" features. Similar approximate inference techniques support efficient parameter estimation with hidden variables. We use the decoder to conduct controlled experiments on a German-to-English translation task, to compare lexical phrase, syntax, and combined models, and to measure effects of various restrictions on non-isomorphism.

1 Introduction

We have seen rapid recent progress in machine translation through the use of rich features and the development of improved decoding algorithms, often based on grammatical formalisms.[1] If we view MT as a machine learning problem, features and formalisms imply structural independence assumptions, which are in turn exploited by efficient inference algorithms, including decoders (Koehn et al., 2003; Yamada and Knight, 2001). Hence a tension is visible in the many recent research efforts aiming to decode with "non-local" features (Chiang, 2007; Huang and Chiang, 2007).

Lopez (2009) recently argued for a separation between features/formalisms (and the independence assumptions they imply) from inference algorithms in MT; this separation is widely appreciated in machine learning. Here we take first steps toward such a "universal" decoder, making the following contributions:

Arbitrary feature model (§2): We define a single, direct log-linear translation model (Papineni et al., 1997; Och and Ney, 2002) that encodes most popular MT features and can be used to encode *any* features on source and target sentences, dependency trees, and alignments. The trees are optional and can be easily removed, allowing simulation of "string-to-tree," "tree-to-string," "tree-to-tree," and "phrase-based" models, among many others. We follow the widespread use of log-linear modeling for direct translation modeling; the novelty is in the use of richer feature sets than have been previously used in a single model.

Decoding as QG parsing (§3–4): We present a novel decoder based on lattice parsing with quasi-synchronous grammar (QG; Smith and Eisner, 2006).[2] Further, we exploit generic approximate inference techniques to incorporate arbitrary "non-local" features in the dynamic programming algorithm (Chiang, 2007; Gimpel and Smith, 2009).

Parameter estimation (§5): We exploit similar approximate inference methods in regularized pseudolikelihood estimation (Besag, 1975) with hidden variables to discriminatively and efficiently train our model. Because we start with *inference* (the key subroutine in training), many other learning algorithms are possible.

Experimental platform (§6): The flexibility of our model/decoder permits carefully controlled experiments. We compare lexical phrase and dependency syntax features, as well as a novel com-

[1]Informally, features are "parts" of a parallel sentence pair and/or their mutual derivation structure (trees, alignments, etc.). Features are often implied by a choice of formalism.

[2]To date, QG has been used for word alignment (Smith and Eisner, 2006), adaptation and projection in parsing (Smith and Eisner, 2009), and various monolingual recognition and scoring tasks (Wang et al., 2007; Das and Smith, 2009); this paper represents its first application to MT.

Σ, T	source and target language vocabularies, respectively
$\mathrm{Trans} : \Sigma \cup \{\textsc{null}\} \rightarrow 2^{\mathrm{T}}$	function mapping each source word to target words to which it may translate
$\boldsymbol{s} = \langle s_0, \ldots, s_n \rangle \in \Sigma^n$	source language sentence (s_0 is the NULL word)
$\boldsymbol{t} = \langle t_1, \ldots, t_m \rangle \in \mathrm{T}^m$	target language sentence, translation of $\boldsymbol{s}$
$\tau_{\boldsymbol{s}} : \{1, \ldots, n\} \rightarrow \{0, \ldots, n\}$	dependency tree of $\boldsymbol{s}$, where $\tau_{\boldsymbol{s}}(i)$ is the index of the parent of s_i (0 is the root, \$)
$\tau_{\boldsymbol{t}} : \{1, \ldots, m\} \rightarrow \{0, \ldots, m\}$	dependency tree of $\boldsymbol{t}$, where $\tau_{\boldsymbol{t}}(i)$ is the index of the parent of t_i (0 is the root, \$)
$\boldsymbol{a} : \{1, \ldots, m\} \rightarrow 2^{\{1, \ldots, n\}}$	alignments from words in $\boldsymbol{t}$ to words in $\boldsymbol{s}$; $\emptyset$ denotes alignment to NULL
$\boldsymbol{\theta}$	parameters of the model
$\boldsymbol{g}_{trans}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{t})$	*lexical translation features (§2.1):*
$\boldsymbol{f}_{lex}(s, t)$	word-to-word translation features for translating s as t
$\boldsymbol{f}_{phr}(\boldsymbol{s}_i^j, \boldsymbol{t}_k^\ell)$	phrase-to-phrase translation features for translating $\boldsymbol{s}_i^j$ as $\boldsymbol{t}_k^\ell$
$\boldsymbol{g}_{lm}(\boldsymbol{t})$	*language model features (§2.2):*
$\boldsymbol{f}_N(\boldsymbol{t}_{j-N+1}^j)$	N-gram probabilities
$\boldsymbol{g}_{syn}(\boldsymbol{t}, \tau_{\boldsymbol{t}})$	*target syntactic features (§2.3):*
$\boldsymbol{f}_{att}(t, j, t', k)$	syntactic features for attaching target word t' at position k to target word t at position j
$\boldsymbol{f}_{val}(t, j, I)$	syntactic valence features with word t at position j having children $I \subseteq \{1, \ldots, m\}$
$\boldsymbol{g}_{reor}(\boldsymbol{s}, \tau_{\boldsymbol{s}}, \boldsymbol{a}, \boldsymbol{t}, \tau_{\boldsymbol{t}})$	*reordering features (§2.4):*
$\boldsymbol{f}_{dist}(i, j)$	distortion features for a source word at position i aligned to a target word at position j
$\boldsymbol{g}_{tree^2}(\tau_{\boldsymbol{s}}, \boldsymbol{a}, \tau_{\boldsymbol{t}})$	*tree-to-tree syntactic features (§3):*
$\boldsymbol{f}_{qg}(i, i', j, k)$	configuration features for source pair $s_i/s_{i'}$ being aligned to target pair t_j/t_k
$\boldsymbol{g}_{cov}(\boldsymbol{a})$	*coverage features (§4.2)*
$\boldsymbol{f}_{\boldsymbol{s}cov}(\boldsymbol{a}), \boldsymbol{f}_{z\mathrm{th}}(\boldsymbol{a}), \boldsymbol{f}_{\boldsymbol{s}unc}(\boldsymbol{a})$	counters for "covering" each $\boldsymbol{s}$ word each time, the zth time, and leaving it "uncovered"

Table 1: Key notation. Feature factorings are elaborated in Tab. 2.

bination of the two. We quantify the effects of our approximate inference. We explore the effects of various ways of restricting syntactic non-isomorphism between source and target trees through the QG. We do not report state-of-the-art performance, but these experiments reveal interesting trends that will inform continued research.

2 Model

(Table 1 explains notation.) Given a sentence $\boldsymbol{s}$ and its parse tree $\tau_{\boldsymbol{s}}$, we formulate the translation problem as finding the target sentence $\boldsymbol{t}^*$ (along with its parse tree $\tau_{\boldsymbol{t}}^*$ and alignment $\boldsymbol{a}^*$ to the source tree) such that[3]

$$\langle \boldsymbol{t}^*, \tau_{\boldsymbol{t}}^*, \boldsymbol{a}^* \rangle = \underset{\langle \boldsymbol{t}, \tau_{\boldsymbol{t}}, \boldsymbol{a} \rangle}{\operatorname{argmax}}\; p(\boldsymbol{t}, \tau_{\boldsymbol{t}}, \boldsymbol{a} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}}) \quad (1)$$

In order to include overlapping features and permit hidden variables during training, we use a single globally-normalized conditional log-linear model. That is, $p(\boldsymbol{t}, \tau_{\boldsymbol{t}}, \boldsymbol{a} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}}) =$

$$\frac{\exp\{\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}, \tau_{\boldsymbol{s}}, \boldsymbol{a}, \boldsymbol{t}, \tau_{\boldsymbol{t}})\}}{\sum_{\boldsymbol{a}', \boldsymbol{t}', \tau_{\boldsymbol{t}}'} \exp\{\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}, \tau_{\boldsymbol{s}}, \boldsymbol{a}', \boldsymbol{t}', \tau_{\boldsymbol{t}}')\}} \quad (2)$$

where the $\boldsymbol{g}$ are arbitrary feature functions and the $\boldsymbol{\theta}$ are feature weights. If one or both parse trees or the word alignments are unavailable, they can be ignored or marginalized out as hidden variables.

In a log-linear model over structured objects, the choice of feature functions $\boldsymbol{g}$ has a huge effect on the feasibility of inference, including decoding. Typically these feature functions are chosen to *factor* into local parts of the overall structure. We next define some key features used in current MT systems, explaining how they factor. We will use subscripts on $\boldsymbol{g}$ to denote different groups of features, which may depend on subsets of the structures $\boldsymbol{t}$, $\tau_{\boldsymbol{t}}$, $\boldsymbol{a}$, $\boldsymbol{s}$, and $\tau_{\boldsymbol{s}}$. When these features factor into parts, we will use $\boldsymbol{f}$ to denote the factored vectors, so that if $\boldsymbol{x}$ is an object that breaks into parts $\{x_i\}_i$, then $\boldsymbol{g}(\boldsymbol{x}) = \sum_i \boldsymbol{f}(x_i)$.[4]

2.1 Lexical Translations

Classical lexical translation features depend on $\boldsymbol{s}$ and $\boldsymbol{t}$ and the alignment $\boldsymbol{a}$ between them. The simplest are word-to-word features, estimated as the conditional probabilities $p(t \mid s)$ and $p(s \mid t)$ for $s \in \Sigma$ and $t \in \mathrm{T}$. Phrase-to-phrase features generalize these, estimated as $p(\boldsymbol{t}' \mid \boldsymbol{s}')$ and $p(\boldsymbol{s}' \mid \boldsymbol{t}')$ where $\boldsymbol{s}'$ (respectively, $\boldsymbol{t}'$) is a substring of $\boldsymbol{s}$ ($\boldsymbol{t}$).

A major difference between the phrase features used in this work and those used elsewhere is that we do not assume that phrases segment into disjoint parts of the source and target sentences

[3]We assume in this work that s is parsed. In principle, we might include source-side parsing as part of decoding.

[4]There are two conventional definitions of feature functions. One is to let the range of these functions be *conditional probability* estimates (Och and Ney, 2002). These estimates are usually heuristic and inconsistent (Koehn et al., 2003). An alternative is to instantiate features for different structural patterns (Liang et al., 2006; Blunsom et al., 2008). This offers more expressive power but may require much more training data to avoid overfitting. For this reason, and to keep training fast, we opt for the former convention, though our decoder can handle both, and the factorings we describe are agnostic about this choice.

(Koehn et al., 2003); they can overlap.[5] Additionally, since phrase features can be any function of words and alignments, we permit features that consider phrase pairs in which a target word outside the target phrase aligns to a source word inside the source phrase, as well as phrase pairs with gaps (Chiang, 2005; Ittycheriah and Roukos, 2007).

Lexical translation features factor as in Eq. 3 (Tab. 2). We score all phrase pairs in a sentence pair that pair a target phrase with the smallest source phrase that contains all of the alignments in the target phrase; if $\bigcup_{k:i\leq k\leq j} \boldsymbol{a}(k) = \emptyset$, no phrase feature fires for $\boldsymbol{t}_i^j$.

2.2 N-gram Language Model

N-gram language models have become standard in machine translation systems. For bigrams and trigrams (used in this paper), the factoring is in Eq. 4 (Tab. 2).

2.3 Target Syntax

There have been many features proposed that consider source- and target-language syntax during translation. Syntax-based MT systems often use features on grammar rules, frequently maximum likelihood estimates of conditional probabilities in a probabilistic grammar, but other syntactic features are possible. For example, Quirk et al. (2005) use features involving phrases and source-side dependency trees and Mi et al. (2008) use features from a forest of parses of the source sentence. There is also substantial work in the use of target-side syntax (Galley et al., 2006; Marcu et al., 2006; Shen et al., 2008). In addition, researchers have recently added syntactic features to phrase-based and hierarchical phrase-based models (Gimpel and Smith, 2008; Haque et al., 2009; Chiang et al., 2008).

In this work, we focus on syntactic features of target-side dependency trees, $\tau_{\boldsymbol{t}}$, along with the words $\boldsymbol{t}$. These include attachment features that relate a word to its syntactic parent, and valence features. They factor as in Eq. 5 (Tab. 2). Features that consider only target-side syntax and words without considering $\boldsymbol{s}$ can be seen as "syntactic language model" features (Shen et al., 2008).

[5]Segmentation might be modeled as a hidden variable in future work.

$$\begin{aligned}
\boldsymbol{g}_{trans}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{t}) &= \textstyle\sum_{j=1}^{m}\sum_{i\in\boldsymbol{a}(j)} \boldsymbol{f}_{lex}(s_i,t_j) && (3)\\
&\quad + \textstyle\sum_{i,j:1\leq i<j\leq m} \boldsymbol{f}_{phr}(\boldsymbol{s}_{\mathit{first}(i,j)}^{last(i,j)}, \boldsymbol{t}_i^j) &&\\
\boldsymbol{g}_{lm}(\boldsymbol{t}) &= \textstyle\sum_{N\in\{2,3\}}\sum_{j=1}^{m+1} \boldsymbol{f}_N(\boldsymbol{t}_{j-N+1}^j) && (4)\\
\boldsymbol{g}_{syn}(\boldsymbol{t},\tau_{\boldsymbol{t}}) &= \textstyle\sum_{j=1}^{m} \boldsymbol{f}_{att}(t_j, j, t_{\tau_{\boldsymbol{t}}(j)}, \tau_{\boldsymbol{t}}(j)) &&\\
&\quad + \boldsymbol{f}_{val}(t_j, j, \tau_{\boldsymbol{t}}^{-1}(j)) && (5)\\
\boldsymbol{g}_{reor}(\boldsymbol{s},\tau_{\boldsymbol{s}},\boldsymbol{a},\boldsymbol{t},\tau_{\boldsymbol{t}}) &= \textstyle\sum_{j=1}^{m}\sum_{i\in\boldsymbol{a}(j)} \boldsymbol{f}_{dist}(i,j) && (6)\\
\boldsymbol{g}_{tree^2}(\tau_{\boldsymbol{s}},\boldsymbol{a},\tau_{\boldsymbol{t}}) &= \sum_{j=1}^{m} \boldsymbol{f}_{qg}(\boldsymbol{a}(j), \boldsymbol{a}(\tau_{\boldsymbol{t}}(j)), j, \tau_{\boldsymbol{t}}(j)) && (7)
\end{aligned}$$

Table 2: Factoring of global feature collections $\boldsymbol{g}$ into $\boldsymbol{f}$. $\boldsymbol{x}_i^j$ denotes $\langle x_i, \ldots x_j\rangle$ in sequence $\boldsymbol{x} = \langle x_1, \ldots\rangle$. $\mathit{first}(i,j) = \min_{k:i\leq k\leq j}(\min(\boldsymbol{a}(k)))$ and $last(i,j) = \max_{k:i\leq k\leq j}(\max(\boldsymbol{a}(k)))$.

2.4 Reordering

Reordering features take many forms in MT. In phrase-based systems, reordering is accomplished both within phrase pairs (local reordering) as well as through distance-based distortion models (Koehn et al., 2003) and lexicalized reordering models (Koehn et al., 2007). In syntax-based systems, reordering is typically parameterized by grammar rules. For generality we permit these features to "see" all structures and denote them $\boldsymbol{g}_{reor}(\boldsymbol{s},\tau_{\boldsymbol{s}},\boldsymbol{a},\boldsymbol{t},\tau_{\boldsymbol{t}})$. Eq. 6 (Tab. 2) shows a factoring of reordering features based on absolute positions of aligned words.

We turn next to the "backbone" model for our decoder; the formalism and the properties of its decoding algorithm will inspire two additional sets of features.

3 Quasi-Synchronous Grammars

A *quasi-synchronous dependency grammar* (QDG; Smith and Eisner, 2006) specifies a conditional model $p(\boldsymbol{t},\tau_{\boldsymbol{t}},\boldsymbol{a} \mid \boldsymbol{s},\tau_{\boldsymbol{s}})$. Given a source sentence $\boldsymbol{s}$ and its parse $\tau_{\boldsymbol{s}}$, a QDG induces a probabilistic monolingual dependency 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 $\boldsymbol{s}$. Each word generated by $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$ is annotated with a "sense," which consists of zero or more words from $\boldsymbol{s}$. The senses imply an alignment ($\boldsymbol{a}$) between words in $\boldsymbol{t}$ and words in $\boldsymbol{s}$, or equivalently, between nodes in $\tau_{\boldsymbol{t}}$ and nodes in $\tau_{\boldsymbol{s}}$. In principle, any portion of $\tau_{\boldsymbol{t}}$ may align to any portion of $\tau_{\boldsymbol{s}}$, but in practice we often make restrictions on the alignments to simplify computation. Smith and Eisner, for example, restricted $|\boldsymbol{a}(j)|$ for all words

t_j to be at most one, so that each target word aligned to at most one source word, which we also do here.[6]

Which translations are possible depends heavily on the *configurations* that the QDG permits. Formally, for a parent-child pair $\langle t_{\tau_{\boldsymbol{t}}(j)}, t_j \rangle$ in $\tau_{\boldsymbol{t}}$, we consider the relationship between $\boldsymbol{a}(\tau_{\boldsymbol{t}}(j))$ and $\boldsymbol{a}(j)$, the source-side words to which $t_{\tau_{\boldsymbol{t}}(j)}$ and t_j align. If, for example, we require that, for all j, $\boldsymbol{a}(\tau_{\boldsymbol{t}}(j)) = \tau_{\boldsymbol{s}}(\boldsymbol{a}(j))$ or $\boldsymbol{a}(j) = 0$, and that the root of $\tau_{\boldsymbol{t}}$ must align to the root of $\tau_{\boldsymbol{s}}$ or to NULL, then strict isomorphism must hold between $\tau_{\boldsymbol{s}}$ and $\tau_{\boldsymbol{t}}$, and we have implemented a synchronous CF dependency grammar (Alshawi et al., 2000; Ding and Palmer, 2005). Smith and Eisner (2006) grouped all possible configurations into eight classes and explored the effects of permitting different sets of classes in word alignment. ("$\boldsymbol{a}(\tau_{\boldsymbol{t}}(j)) = \tau_{\boldsymbol{s}}(\boldsymbol{a}(j))$" corresponds to their "parent-child" configuration; see Fig. 3 in Smith and Eisner (2006) for illustrations of the rest.) More generally, we can define *features* on tree pairs that factor into these local configurations, as shown in Eq. 7 (Tab. 2).

Note that the QDG instantiates the model in Eq. 2. Of the features discussed in §2, $\boldsymbol{f}_{lex}$, $\boldsymbol{f}_{att}$, $\boldsymbol{f}_{val}$, and $\boldsymbol{f}_{dist}$ can be easily incorporated into the QDG as described while respecting the independence assumptions implied by the configuration features. The others ($\boldsymbol{f}_{phr}$, $\boldsymbol{f}_2$, and $\boldsymbol{f}_3$) are *non-local*, or involve parts of the structure that, from the QDG's perspective, are conditionally independent given intervening material. Note that "non-locality" is relative to a choice of formalism; in §2 we did not commit to any formalism, so it is only now that we can describe phrase and N-gram features as non-local. Non-local features will present a challenge for decoding and training (§4.3).

4 Decoding

Given a sentence $\boldsymbol{s}$ and its parse $\tau_{\boldsymbol{s}}$, at decoding time we seek the target sentence $\boldsymbol{t}^*$, the target tree $\tau_{\boldsymbol{t}}^*$, and the alignments $\boldsymbol{a}^*$ that are most probable, as defined in Eq. 1.[7] (In §5 we will consider k-best and all-translations variations on this problem.) As usual, the normalization constant is not required for decoding; it suffices to solve:

$$\langle \boldsymbol{t}^*, \tau_{\boldsymbol{t}}^*, \boldsymbol{a}^* \rangle = \operatorname*{argmax}_{\langle \boldsymbol{t}, \tau_{\boldsymbol{t}}, \boldsymbol{a} \rangle} \boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}, \tau_{\boldsymbol{s}}, \boldsymbol{a}, \boldsymbol{t}, \tau_{\boldsymbol{t}}) \quad (8)$$

For a QDG model, the decoding problem has not been addressed before. It equates to finding the most probable derivation under the $\boldsymbol{s}/\tau_{\boldsymbol{s}}$-specific grammar $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$. We solve this by lattice parsing, assuming that an upper bound on m (the length of $\boldsymbol{t}$) is known. The advantage offered by this approach (like most other grammar-based translation approaches) is that decoding becomes *dynamic programming* (DP), a technique that is both widely understood in NLP and for which practical, efficient, generic techniques exist. A major advantage of DP is that, with small modifications, *summing* over structures is also possible with "inside" DP algorithms. We will exploit this in training (§5). Efficient summing opens up many possibilities for training $\boldsymbol{\theta}$, such as likelihood and pseudo-likelihood, and provides principled ways to handle hidden variables during learning.

4.1 Translation as Monolingual Parsing

We decode by performing lattice parsing on a lattice encoding the set of possible translations. The lattice is a weighted "sausage" lattice that permits sentences up to some maximum length ℓ; ℓ is derived from the source sentence length. Let the states be numbered 0 to ℓ; states from $\lfloor \rho\ell \rfloor$ to ℓ are final states (for some $\rho \in (0, 1)$). For every position between consecutive states $j - 1$ and j $(0 < j \leq \ell)$, and for every word s_i in $\boldsymbol{s}$, and for every word $t \in \text{Trans}(s_i)$, we instantiate an arc annotated with t and i. The weight of such an arc is $\exp\{\boldsymbol{\theta}^\top \boldsymbol{f}\}$, where $\boldsymbol{f}$ is the sum of feature functions that fire when s_i translates as t in target position j (e.g., $\boldsymbol{f}_{lex}(s_i, t)$ and $\boldsymbol{f}_{dist}(i, j)$).

Given the lattice and $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$, lattice parsing is a straightforward generalization of standard context-free dependency parsing DP algorithms (Eisner, 1997). This decoder accounts for $\boldsymbol{f}_{lex}$, $\boldsymbol{f}_{att}$, $\boldsymbol{f}_{val}$, $\boldsymbol{f}_{dist}$, and $\boldsymbol{f}_{qg}$ as local features.

Figure 1 gives an example, showing a German sentence and dependency tree from an automatic parser, an English reference, and a lattice representing possible translations. In each bundle, the arcs are listed in decreasing order according to weight and for clarity only the first five are shown. The output of the decoder consists of lattice arcs

[6]I.e., from here on, $\boldsymbol{a} : \{1, \ldots, m\} \rightarrow \{0, \ldots, n\}$ where 0 denotes alignment to NULL.

[7]Arguably, we seek $\text{argmax}_{\boldsymbol{t}}\, p(\boldsymbol{t} \mid \boldsymbol{s})$, marginalizing out everything else. Approximate solutions have been proposed for that problem in several settings (Blunsom and Osborne, 2008; Sun and Tsujii, 2009); we leave their combination with our approach to future work.

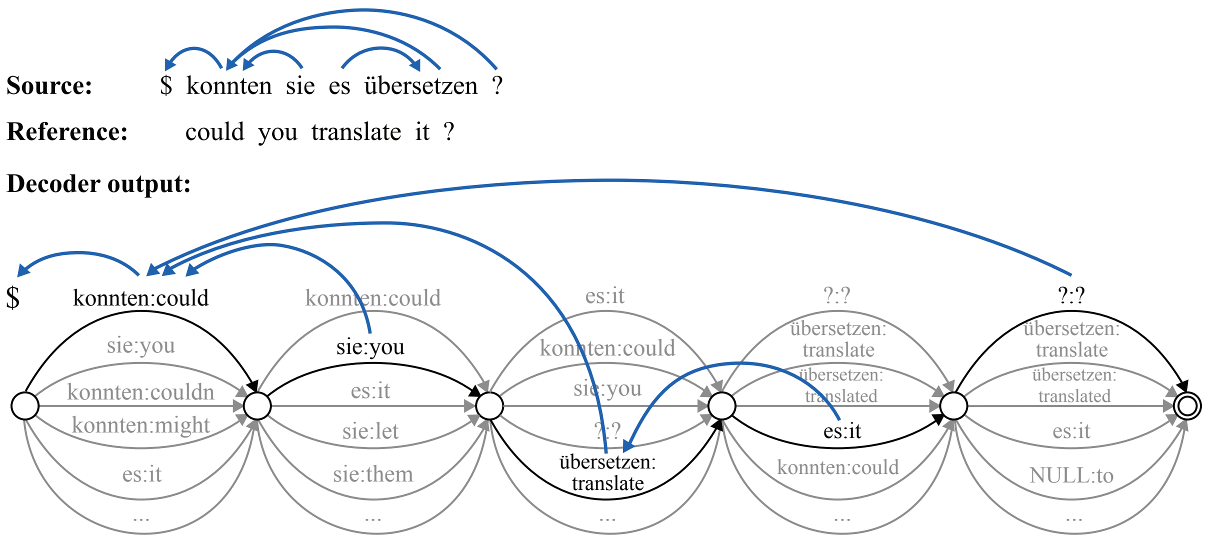

Figure 1: Decoding as lattice parsing, with the highest-scoring translation denoted by black lattice arcs (others are grayed out) and thicker blue arcs forming a dependency tree over them.

selected at each position and a dependency tree over them.

4.2 Source-Side Coverage Features

Most MT decoders enforce a notion of "coverage" of the source sentence during translation: all parts of $\boldsymbol{s}$ should be aligned to some part of $\boldsymbol{t}$ (alignment to NULL incurs an explicit cost). Phrase-based systems such as Moses (Koehn et al., 2007) explicitly search for the highest-scoring string in which all source words are translated. Systems based on synchronous grammars proceed by parsing the source sentence with the synchronous grammar, ensuring that every phrase and word has an analogue in $\tau_{\boldsymbol{t}}$ (or a deliberate choice is made by the decoder to translate it to NULL). In such systems, we do not need to use features to implement source-side coverage, as it is assumed as a hard constraint always respected by the decoder.

Our QDG decoder has no way to enforce coverage; it does not track any kind of state in $\tau_{\boldsymbol{s}}$ apart from a single recently aligned word. This is a problem with other direct translation models, such as IBM model 1 used as a direct model rather than a channel model (Brown et al., 1993). This sacrifice is the result of our choice to use a conditional model (§2).

The solution is to introduce a set of *coverage* features $\boldsymbol{g}_{cov}(\boldsymbol{a})$. Here, these include:

- A counter for the number of times each source word is covered: $\boldsymbol{f}_{\boldsymbol{s}cov}(\boldsymbol{a}) = \sum_{i=1}^{n} |\boldsymbol{a}^{-1}(i)|$.
- Features that fire once when a source word is covered the zth time ($z \in \{2, 3, 4\}$) and fire again all subsequent times it is covered; these are denoted $\boldsymbol{f}_{2\text{nd}}$, $\boldsymbol{f}_{3\text{rd}}$, and $\boldsymbol{f}_{4\text{th}}$.
- A counter of uncovered source words: $\boldsymbol{f}_{\boldsymbol{s}unc}(\boldsymbol{a}) = \sum_{i=1}^{n} \delta(|\boldsymbol{a}^{-1}(i)|, 0)$.

Of these, only $\boldsymbol{f}_{\boldsymbol{s}cov}$ is local.

4.3 Non-Local Features

The lattice QDG parsing decoder incorporates many of the features we have discussed, but not all of them. Phrase lexicon features $\boldsymbol{f}_{phr}$, language model features $\boldsymbol{f}_N$ for $N > 1$, and most coverage features are non-local with respect to our QDG. Recently Chiang (2007) introduced "cube pruning" as an approximate decoding method that extends a DP decoder with the ability to incorporate features that break the Markovian independence assumptions DP exploits. Techniques like cube pruning can be used to include the non-local features in our decoder.[8]

5 Training

Training requires us to learn values for the parameters $\boldsymbol{\theta}$ in Eq. 2. Given T training examples of the form $\langle \boldsymbol{t}^{(i)}, \tau_{\boldsymbol{t}}^{(i)}, \boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)} \rangle$, for $i = 1, ..., T$, maximum likelihood estimation for this model consists of solving Eq. 9 (Tab. 3).[9] Note that the

[8]A full discussion is omitted for space, but in fact we use "cube decoding," a slightly less approximate, slightly more expensive method that is more closely related to the approximate inference methods we use for training, discussed in §5.

[9]In practice, we regularize by including a term $-c\|\boldsymbol{\theta}\|_2^2$.

$$\mathrm{LL}(\boldsymbol{\theta}) = \sum_{i=1}^{T} \log p(\boldsymbol{t}^{(i)}, \tau_{\boldsymbol{t}}^{(i)} \mid \boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)}) = \sum_{i=1}^{T} \log \frac{\sum_{\boldsymbol{a}} \exp\{\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)}, \boldsymbol{a}, \boldsymbol{t}^{(i)}, \tau_{\boldsymbol{t}}^{(i)})\}}{\sum_{\boldsymbol{t}, \tau_{\boldsymbol{t}}, \boldsymbol{a}} \exp\{\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)}, \boldsymbol{a}, \boldsymbol{t}, \tau_{\boldsymbol{t}})\}} = \sum_{i=1}^{T} \log \frac{\text{“numerator”}}{\text{“denominator”}} \tag{9}$$

$$\mathrm{PL}(\boldsymbol{\theta}) = \sum_{i=1}^{T} \log\Big(\sum_{\boldsymbol{a}} p(\boldsymbol{t}^{(i)}, \boldsymbol{a} \mid \tau_{\boldsymbol{t}}^{(i)}, \boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)})\Big) + \sum_{i=1}^{T} \log\Big(\sum_{\boldsymbol{a}} p(\tau_{\boldsymbol{t}}^{(i)}, \boldsymbol{a} \mid \boldsymbol{t}^{(i)}, \boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)})\Big) \tag{10}$$

$$\begin{array}{l}\text{“denominator” of}\\ \text{term 1 in Eq. 10}\end{array} = \sum_{i=0}^{n} \sum_{t' \in \mathrm{Trans}(s_i)} S(\tau_{\boldsymbol{t}}^{-1}(0), i, t') \times \exp\left\{\boldsymbol{\theta}^\top \left(\boldsymbol{f}_{lex}(s_i, t') + \boldsymbol{f}_{att}(\$, 0, t', k) + \boldsymbol{f}_{qg}(0, i, 0, k)\right)\right\} \tag{11}$$

$$S(j, i, t) = \prod_{k \in \tau_{\boldsymbol{t}}^{-1}(j)} \sum_{i'=0}^{n} \sum_{t' \in \mathrm{Trans}(s_{i'})} S(k, i', t') \times \exp\left\{\boldsymbol{\theta}^\top \left(\begin{array}{l} \boldsymbol{f}_{lex}(s_{i'}, t') + \boldsymbol{f}_{att}(t, j, t', k) + \\ \boldsymbol{f}_{val}(t, j, \tau_{\boldsymbol{t}}^{-1}(j)) + \boldsymbol{f}_{qg}(i, i', j, k) \end{array}\right)\right\} \tag{12}$$

$$S(j, i, t) = \exp\left\{\boldsymbol{\theta}^\top \left(\boldsymbol{f}_{val}(t, j, \tau_{\boldsymbol{t}}^{-1}(j))\right)\right\} \quad \text{if } \tau_{\boldsymbol{t}}^{-1}(j) = \emptyset \tag{13}$$

Table 3: Eq. 9: Log-likelihood. Eq. 10: Pseudolikelihood. In both cases we maximize w.r.t. $\boldsymbol{\theta}$. Eqs. 11–13: Recursive DP equations for summing over $\boldsymbol{t}$ and $\boldsymbol{a}$.

alignments are treated as a hidden variable to be marginalized out.[10] Optimization problems of this form are by now widely known in NLP (Koo and Collins, 2005), and have recently been used for machine translation as well (Blunsom et al., 2008). Such problems are typically solved using variations of gradient ascent; in our experiments, we will use an online method called stochastic gradient ascent (SGA). This requires us to calculate the function's gradient (vector of first derivatives) with respect to $\boldsymbol{\theta}$.[11]

Computing the numerator in Eq. 9 involves summing over all possible alignments; with QDG and a hard bound of 1 on $|\boldsymbol{a}(j)|$ for all j, a fast “inside” DP solution is known (Smith and Eisner, 2006; Wang et al., 2007). It runs in $O(mn^2)$ time and $O(mn)$ space.

Computing the denominator in Eq. 9 requires summing over all word sequences and dependency trees for the target language sentence and all word alignments between the sentences. With a maximum length imposed, this is tractable using the “inside” version of the maximizing DP algorithm of Sec. 4, but it is prohibitively expensive. We therefore optimize *pseudo-likelihood* instead, making the following approximation (Besag, 1975):

$$p(\boldsymbol{t}, \tau_{\boldsymbol{t}} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}}) \approx p(\boldsymbol{t} \mid \tau_{\boldsymbol{t}}, \boldsymbol{s}, \tau_{\boldsymbol{s}}) \times p(\tau_{\boldsymbol{t}} \mid \boldsymbol{t}, \boldsymbol{s}, \tau_{\boldsymbol{s}})$$

Plugging this into Eq. 9, we arrive at Eq. 10 (Tab. 3). The two parenthesized terms in Eq. 10 each have their own numerators and denominators (not shown). The numerators are identical to each other and to that in Eq. 9. The denominators are much more manageable than in Eq. 9, never requiring summation over more than two structures at a time. We must sum over target word sequences and word alignments (with fixed $\tau_{\boldsymbol{t}}$), and separately over target trees and word alignments (with fixed $\boldsymbol{t}$).

5.1 Summing over $\boldsymbol{t}$ and $\boldsymbol{a}$

The summation over target word sequences and alignments given fixed $\tau_{\boldsymbol{t}}$ bears a resemblance to the inside algorithm, except that the tree structure is fixed (Pereira and Schabes, 1992). Let $S(j, i, t)$ denote the sum of all translations rooted at position j in $\tau_{\boldsymbol{t}}$ such that $\boldsymbol{a}(j) = i$ and $t_j = t$.

Tab. 3 gives the equations for this DP: Eq. 11 is the quantity of interest, Eq. 12 is the recursion, and Eq. 13 shows the base cases for leaves of $\tau_{\boldsymbol{t}}$.

Letting $q = \max_{0 \le i \le n} |\mathrm{Trans}(s_i)|$, this algorithm runs in $O(mn^2q^2)$ time and $O(mnq)$ space. For efficiency we place a hard upper bound on q during training (details in §6).

5.2 Summing over $\tau_{\boldsymbol{t}}$ and $\boldsymbol{a}$

For the summation over dependency trees and alignments given fixed $\boldsymbol{t}$, required for $p(\tau_{\boldsymbol{t}} \mid \boldsymbol{t}, \boldsymbol{s}, \tau_{\boldsymbol{s}})$, we perform “inside” lattice parsing with $G_{\boldsymbol{s}, \tau_{\boldsymbol{s}}}$. The technique is the summing variant of the decoding method in §4, except for each state j,

[10]Alignments could be supplied by automatic word alignment algorithms. We chose to leave them hidden so that we could make the best use of our parsed training data when configuration constraints are imposed, since it is not always possible to reconcile automatic word alignments with automatic parses.

[11]When the function's value is computed by “inside” DP, the corresponding “outside” algorithm can be used to obtain the gradient. Because outside algorithms can be automatically derived from inside ones, we discuss only inside algorithms in this paper; see Eisner et al. (2005).

the sausage lattice only includes arcs from $j-1$ to j that are labeled with the known target word t_j. If a is the number of arcs in the lattice, which is $O(mn)$, this algorithm runs in $O(a^3)$ time and requires $O(a^2)$ space. Because we use a hard upper bound on $|\mathrm{Trans}(s)|$ for all $s \in \Sigma$, this summation is much faster in practice than the one over words and alignments.

5.3 Handling Non-Local Features

So far, all of our algorithms have exploited DP, disallowing any *non-local* features (e.g., $\boldsymbol{f}_{phr}$, $\boldsymbol{f}_N$ for $N > 1$, $\boldsymbol{f}_{z\mathrm{th}}$, $\boldsymbol{f}_{\boldsymbol{s}unc}$). We recently proposed "cube summing," an approximate technique that permits the use of non-local features for inside DP algorithms (Gimpel and Smith, 2009). Cube summing is based on a slightly less greedy variation of cube *pruning* (Chiang, 2007) that maintains k-best lists of derivations for each DP chart item. Cube summing augments the k-best list with a residual term that sums over remaining structures not in the k-best list, albeit without their non-local features. Using the machinery of cube summing, it is straightforward to include the desired non-local features in the summations required for pseudo-likelihood, as well as to compute their approximate gradients.

Our approach permits an alternative to minimum error-rate training (MERT; Och, 2003); it is discriminative but handles latent structure and regularization in more principled ways. The pseudo-likelihood calculations for a sentence pair, taken together, are faster than (k-best) decoding, making SGA's inner loop faster than MERT's inner loop.

6 Experiments

Our decoding framework allows us to perform many experiments with the same feature representation and inference algorithms, including combining and comparing phrase-based and syntax-based features and examining how isomorphism constraints of synchronous formalisms affect translation output.

6.1 Data and Evaluation

We use the German-English portion of the Basic Travel Expression Corpus (BTEC). The corpus has approximately 100K sentence pairs. We filter sentences of length more than 15 words, which only removes 6% of the data. We end up with a training set of 82,299 sentences, a development set of 934 sentences, and a test set of 500 sentences. We evaluate translation output using case-insensitive BLEU (Papineni et al., 2001), as provided by NIST, and METEOR (Banerjee and Lavie, 2005), version 0.6, with Porter stemming and WordNet synonym matching.

6.2 Features

Our base system uses features as discussed in §2. To obtain lexical translation features $\boldsymbol{g}_{trans}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{t})$, we use the Moses pipeline (Koehn et al., 2007). We perform word alignment using GIZA++ (Och and Ney, 2003), symmetrize the alignments using the "grow-diag-final-and" heuristic, and extract phrases up to length 3. We define $\boldsymbol{f}_{lex}$ by the lexical probabilities $p(t \mid s)$ and $p(s \mid t)$ estimated from the symmetrized alignments. After discarding phrase pairs with only one target-side word (since we only allow a target word to align to at most one source word), we define $\boldsymbol{f}_{phr}$ by 8 features: $\{2, 3\}$ target words $\times$ phrase conditional and "lexical smoothing" probabilities $\times$ two conditional directions.

Bigram and trigam language model features, $\boldsymbol{f}_2$ and $\boldsymbol{f}_3$, are estimated using the SRI toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1998).

For our target-language syntactic features $\boldsymbol{g}_{syn}$, we use features similar to lexicalized CFG events (Collins, 1999), specifically following the dependency model of Klein and Manning (2004). These include probabilities associated with individual attachments ($\boldsymbol{f}_{att}$) and child-generation valence probabilities ($\boldsymbol{f}_{val}$). These probabilities are estimated on the training corpus parsed using the Stanford factored parser (Klein and Manning, 2003). The same probabilities are also included using 50 hard word classes derived from the parallel corpus using the GIZA++ `mkcls` utility (Och and Ney, 2003). In total, there are 7 lexical and 7 word-class syntax features.

For reordering, we use a single absolute distortion feature $\boldsymbol{f}_{dist}(i, j)$ that returns $|i-j|$ whenever $\boldsymbol{a}(j) = i$ and $i, j > 0$. (Unlike the other feature functions, which returned probabilities, this feature function returns a nonnegative integer.)

The tree-to-tree syntactic features $\boldsymbol{g}_{tree^2}$ in our model are binary features $\boldsymbol{f}_{qg}$ that fire for particular QG configurations. We use one feature for each of the configurations in (Smith and Eisner, 2006), adding 7 additional features that score configura-

Phrase features:	Syntactic Features: (base)	$+\boldsymbol{f}_{att} \cup \boldsymbol{f}_{val}$ (target)	$+\boldsymbol{f}_{qg}$ (tree-to-tree)
(base)	0.3727	0.4458	0.4424
$+\boldsymbol{f}_{phr}$	0.4682	0.4971	0.5142

Table 4: Feature set comparison (BLEU).

tions involving root words and NULL-alignments more finely. There are 14 features in this category.

Coverage features $\boldsymbol{g}_{cov}$ are as described in §4.2.

In all, 46 feature weights are learned.

6.3 Experimental Procedure

Our model permits training the system on the full set of parallel data, but we instead use the parallel data to estimate feature functions and learn $\boldsymbol{\theta}$ on the development set.[12] We trained using three iterations of SGA over the development data with a batch size of 1 and a fixed step size of 0.01. We used ℓ_2 regularization with a fixed, untuned coefficient of 0.1. Cube summing used a 10-best list for training and a 7-best list for decoding unless otherwise specified.

To obtain the translation lexicon (Trans) we first included the top three target words t for each s using $p(s \mid t) \times p(t \mid s)$ to score target words. For any training sentence $\langle \boldsymbol{s}, \boldsymbol{t} \rangle$ and t_j for which $t_j \notin \bigcup_{i=1}^{n} \text{Trans}(s_i)$, we added t_j to $\text{Trans}(s_i)$ for $i = \text{argmax}_{i' \in I}\, p(s_{i'}|t_j) \times p(t_j|s_{i'})$, where $I = \{i : 0 \leq i \leq n \wedge |\text{Trans}(s_i)| < q_i\}$. We used $q_0 = 10$ and $q_{>0} = 5$, restricting $|\text{Trans}(\text{NULL})| \leq 10$ and $|\text{Trans}(s)| \leq 5$ for any $s \in \Sigma$. This made 191 of the development sentences unreachable by the model, leaving 743 sentences for learning $\boldsymbol{\theta}$.

During decoding, we generated lattices with all $t \in \text{Trans}(s_i)$ for $0 \leq i \leq n$, for every position. We used $\rho = 0.9$, causing states within 90% of the source sentence length to be final states. Between each pair of consecutive states, we pruned edges that fell outside a beam of 70% of the sum of edge weights (see §4.1; edge weights use $\boldsymbol{f}_{lex}$, $\boldsymbol{f}_{dist}$, and $\boldsymbol{f}_{scov}$) of all edges between those two states.

6.4 Feature Set Comparison

Our first set of experiments compares feature sets commonly used in phrase- and syntax-based translation. In particular, we compare the effects of combining phrase features and syntactic features. The base model contains $\boldsymbol{f}_{lex}$, $\boldsymbol{g}_{lm}$, $\boldsymbol{g}_{reor}$, and $\boldsymbol{g}_{cov}$. The results are shown in Table 4. The second row contains scores when adding in the eight $\boldsymbol{f}_{phr}$ features. The second column shows scores when adding the 14 target syntax features ($\boldsymbol{f}_{att}$ and $\boldsymbol{f}_{val}$), and the third column adds to them the 14 additional tree-to-tree features ($\boldsymbol{f}_{qg}$). We find large gains in BLEU by adding more features, and find that gains obtained through phrase features and syntactic features are partially additive, suggesting that these feature sets are making complementary contributions to translation quality.

6.5 Varying k During Decoding

For models without syntactic features, we constrained the decoder to produce dependency trees in which every word's parent is immediately to its right and ignored syntactic features while scoring structures. This causes decoding to proceed left-to-right in the lattice, the way phrase-based decoders operate. Since these models do not search over trees, they are substantially faster during decoding than those that use syntactic features and do not require any pruning of the lattice. Therefore, we explored varying the value of k used during k-best cube decoding; results are shown in Fig. 2. Scores improve when we increase k up to 10, but not much beyond, and there is still a substantial gap (2.5 BLEU) between using phrase features with $k = 20$ and using all features with $k = 5$. Models without syntax perform poorly when using a very small k, due to their reliance on non-local language model and phrase features. By contrast, models with syntactic features, which are local in our decoder, perform relatively well even with $k = 1$.

6.6 QG Configuration Comparison

We next compare different constraints on isomorphism between the source and target dependency

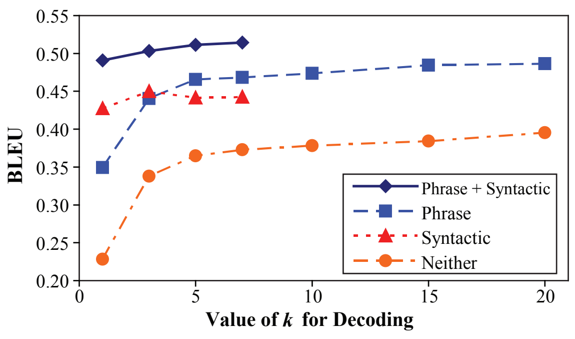

Figure 2: Comparison of size of k-best list for cube decoding with various feature sets.

[12]We made this choice both for similarity to standard MT systems and a more rapid experiment cycle.

QDG Configurations	BLEU	METEOR
synchronous	0.4008	0.6949
+ nulls, root-any	0.4108	0.6931
+ child-parent, same node	0.4337	0.6815
+ sibling	0.4881	0.7216
+ grandparent/child	0.5015	0.7365
+ c-command	0.5156	0.7441
+ other	0.5142	0.7472

Table 5: QG configuration comparison. The name of each configuration, following Smith and Eisner (2006), refers to the relationship between $\boldsymbol{a}(\tau_{\boldsymbol{t}}(j))$ and $\boldsymbol{a}(j)$ in $\tau_{\boldsymbol{s}}$.

trees. To do this, we impose harsh penalties on some QDG configurations (§3) by fixing their feature weights to -1000. Hence they are permitted only when absolutely necessary in training and rarely in decoding.[13] Each model uses all phrase and syntactic features; they differ only in the sets of configurations which have fixed negative weights.

Tab. 5 shows experimental results. The base "synchronous" model permits parent-child ($\boldsymbol{a}(\tau_{\boldsymbol{t}}(j)) = \tau_{\boldsymbol{s}}(\boldsymbol{a}(j))$), any configuration where $\boldsymbol{a}(j) = 0$, including both words being linked to NULL, and requires the root word in $\tau_{\boldsymbol{t}}$ to be linked to the root word in $\tau_{\boldsymbol{s}}$ or to NULL(5 of our 14 configurations). The second row allows any configuration involving NULL, including those where t_j aligns to a non-NULL word in $\boldsymbol{s}$ and its parent aligns to NULL, and allows the root in $\tau_{\boldsymbol{t}}$ to be linked to any word in $\tau_{\boldsymbol{s}}$. Each subsequent row adds additional configurations (i.e., trains its θ rather than fixing it to -1000). In general, we see large improvements as we permit more configurations, and the largest jump occurs when we add the "sibling" configuration ($\tau_{\boldsymbol{s}}(\boldsymbol{a}(\tau_{\boldsymbol{t}}(j))) = \tau_{\boldsymbol{s}}(\boldsymbol{a}(j))$). The BLEU score does not increase, however, when we permit all configurations in the final row of the table, and the METEOR score increases only slightly. While allowing certain categories of non-isomorphism clearly seems helpful, permitting arbitrary violations does not appear to be necessary for this dataset.

6.7 Discussion

We note that these results are not state-of-the-art on this dataset (on this task, Moses/MERT achieves 0.6838 BLEU and 0.8523 METEOR with maximum phrase length 3).[14] Our aim has been to illustrate how a single model can provide a controlled experimental framework for comparisons of features, of inference methods, and of constraints. Our findings show that phrase features and dependency syntax produce complementary improvements to translation quality, that tree-to-tree configurations (a new feature in MT) are helpful for translation, and that substantial gains can be obtained by permitting certain types of non-isomorphism. We have validated cube summing and decoding as practical methods for approximate inference.

Our framework permits exploration of alternative objectives, alternative approximate inference techniques, additional hidden variables (e.g., Moses' phrase segmentation variable), and, of course, additional feature representations. The system is publicly available at `www.ark.cs.cmu.edu/Quipu`.

7 Conclusion

We presented feature-rich MT using a principled probabilistic framework that separates features from inference. Our novel decoder is based on efficient DP-based QG lattice parsing extended to handle "non-local" features using generic techniques that also support efficient parameter estimation. Controlled experiments permitted with this system show interesting trends in the use of syntactic features and constraints.

Acknowledgments

We thank three anonymous EMNLP reviewers, David Smith, and Stephan Vogel for helpful comments and feedback that improved this paper. This research was supported by NSF IIS-0836431 and IIS-0844507, a grant from Google, and computational resources provided by Yahoo.

References

H. Alshawi, S. Bangalore, and S. Douglas. 2000. Learning dependency translation modles as colecctions of finite-state head transducers. *Computational Linguistics*, 26(1):45–60.

S. Banerjee and A. Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*.

J. E. Besag. 1975. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195.

[13] In fact, the strictest "synchronous" model used the almost-forbidden configurations in 2% of test sentences; this behavior disappears as configurations are legalized.

[14] We believe one cause for this performance gap is the generation of the lattice and plan to address this in future work by allowing the phrase table to inform lattice generation.

P. Blunsom and M. Osborne. 2008. Probabilistic inference for machine translation. In *Proc. of EMNLP*.

P. Blunsom, T. Cohn, and M. Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. of ACL*.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

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

D. Chiang, Y. Marton, and P. Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proc. of EMNLP*.

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

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

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, U. Penn.

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

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammar. In *Proc. of ACL*.

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. 1997. Bilexical grammars and a cubic-time probabilistic parser. In *Proc. of IWPT*.

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. 2008. Rich source-side context for statistical machine translation. In *Proc. of ACL-2008 Workshop on Statistical Machine Translation*.

K. Gimpel and N. A. Smith. 2009. Cube summing, approximate inference with non-local features, and dynamic programming without semirings. In *Proc. of EACL*.

R. Haque, S. K. Naskar, Y. Ma, and A. Way. 2009. Using supertags as source language context in SMT. In *Proc. of EAMT*.

L. Huang and D. Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of ACL*.

A. Ittycheriah and S. Roukos. 2007. Direct translation model 2. In *Proc. of HLT-NAACL*.

D. Klein and C. D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in NIPS 15*.

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, 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)*.

T. Koo and M. Collins. 2005. Hidden-variable models for discriminative reranking. In *Proc. of EMNLP*.

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of COLING-ACL*.

A. Lopez. 2009. Translation as weighted deduction. In *Proc. of EACL*.

D. Marcu, W. Wang, A. Echihabi, and K. Knight. 2006. Statistical machine translation with syntactified target language phrases. In *Proc. of EMNLP*.

H. Mi, L. Huang, and Q. Liu. 2008. Forest-based translation. In *Proc. of ACL*.

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, and T. Ward. 1997. Feature-based language understanding. In *EUROSPEECH*.

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

F. C. N. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proc. of ACL*.

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*.

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*.

X. Sun and J. Tsujii. 2009. Sequential labeling with latent variables: An exact inference algorithm and its efficient approximation. In *Proc. of EACL*.

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*.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *Proc. of ACL*.