Hierarchical Phrase-Based Translation

David Chiang[*]
Information Sciences Institute
University of Southern California

We present a statistical machine translation model that uses hierarchical phrases—*phrases that contain subphrases. The model is formally a synchronous context-free grammar but is learned from a parallel text without any syntactic annotations. Thus it can be seen as combining fundamental ideas from both syntax-based translation and phrase-based translation. We describe our system's training and decoding methods in detail, and evaluate it for translation speed and translation accuracy. Using BLEU as a metric of translation accuracy, we find that our system performs significantly better than the Alignment Template System, a state-of-the-art phrase-based system.*

1. Introduction

The alignment template translation model (Och and Ney 2004) and related phrase-based models advanced the state of the art in machine translation by expanding the basic unit of translation from words to **phrases**, that is, substrings of potentially unlimited size (but not necessarily phrases in any syntactic theory). These phrases allow a model to learn local reorderings, translations of multiword expressions, or insertions and deletions that are sensitive to local context. This makes them a simple and powerful mechanism for translation.

The basic phrase-based model is an instance of the noisy-channel approach (Brown et al. 1993). Following convention, we call the source language "French" and the target language "English"; the translation of a French sentence f into an English sentence e is modeled as:

$$\arg\max_e P(e \mid f) = \arg\max_e P(e,f) \tag{1}$$

$$= \arg\max_e (P(e) \times P(f \mid e)) \tag{2}$$

The phrase-based translation model $P(f \mid e)$ "encodes" e into f by the following steps:

1. segment e into phrases $\bar{e}_1 \cdots \bar{e}_I$, typically with a uniform distribution over segmentations;

[*] 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292, USA. E-mail: chiang@isi.edu. Much of the research presented here was carried out while the author was at the University of Maryland Institute for Advanced Computer Studies.

Submission received: 1 May 2006; accepted for publication: 3 October 2006.

2. reorder the $\bar{e}_i$ according to some distortion model;
3. translate each of the $\bar{e}_i$ into French phrases according to a model $P(\bar{f} \mid \bar{e})$ estimated from the training data.

Other phrase-based models model the joint distribution $P(e,f)$ (Marcu and Wong 2002) or make $P(e)$ and $P(f \mid e)$ into features of a log-linear model (Och and Ney 2002). But the basic architecture of phrase segmentation (or generation), phrase reordering, and phrase translation remains the same.

Phrase-based models can robustly perform translations that are localized to substrings that are common enough to have been observed in training. But Koehn, Och, and Marcu (2003) find that phrases longer than three words improve performance little for training corpora of up to 20 million words, suggesting that the data may be too sparse to learn longer phrases. Above the phrase level, some models perform no reordering (Zens and Ney 2004; Kumar, Deng, and Byrne 2006), some have a simple distortion model that reorders phrases independently of their content (Koehn, Och, and Marcu 2003; Och and Ney 2004), and some, for example, the Alignment Template System (Och et al. 2004; Thayer et al. 2004), hereafter ATS, and the IBM phrase-based system (Tillmann 2004; Tillmann and Zhang 2005), have phrase-reordering models that add some lexical sensitivity. But, as an illustration of the limitations of phrase reordering, consider the following Mandarin example and its English translation:

澳洲	是	与	北韩	有	邦交	的	少数	国家	之一	。
Aozhou	shi	yu	Beihan	you	bangjiao	de	shaoshu	guojia	zhiyi	.
Australia	is	with	North Korea	have	dipl. rels.	that	few	countries	one of	.

Australia is one of the few countries that have diplomatic relations with North Korea.

If we count *zhiyi* (literally, 'of-one') as a single token, then translating this sentence correctly into English requires identifying a sequence of five word groups that need to be reversed. When we run a phrase-based system, ATS, on this sentence (using the experimental setup described herein), we get the following phrases with translations:

[Aozhou] [shi]$_1$ [yu Beihan]$_2$ [you] [bangjiao] [de shaoshu guojia zhiyi] [.]

[Australia] [has] [dipl. rels.] [with North Korea]$_2$ [is]$_1$ [one of the few countries] [.]

where we have used subscripts to indicate the reordering of phrases. The phrase-based model is able to order "has diplomatic relations with North Korea" correctly (using phrase reordering) and "is one of the few countries" correctly (using a combination of phrase translation and phrase reordering), but does not invert these two groups as it should.

We propose a solution to these problems that does not interfere with the strengths of the phrase-based approach, but rather capitalizes on them: Because phrases are good for learning reorderings of words, we can use them to learn reorderings of phrases as well. In order to do this we need **hierarchical phrases** that can contain other phrases. For example, a hierarchical phrase pair that might help with the above example is

$$\langle \text{yu } \boxed{1} \text{ you } \boxed{2}, \text{have } \boxed{2} \text{ with } \boxed{1} \rangle \tag{3}$$

where $\boxed{1}$ and $\boxed{2}$ are placeholders for subphrases (Chiang 2005). This would capture the fact that Chinese prepositional phrases almost always modify verb phrases on the left, whereas English prepositional phrases usually modify verb phrases on the right. Because it generalizes over possible prepositional objects and direct objects, it acts both as a discontinuous phrase pair and as a phrase-reordering rule. Thus it is considerably more powerful than a conventional phrase pair.

Similarly, the hierarchical phrase pair

$$\langle \boxed{1} \text{ de } \boxed{2}, \text{the } \boxed{2} \text{ that } \boxed{1} \rangle \tag{4}$$

would capture the fact that Chinese relative clauses modify NPs on the left, whereas English relative clauses modify on the right; and the pair

$$\langle \boxed{1} \text{ zhiyi}, \text{one of } \boxed{1} \rangle \tag{5}$$

would render the construction *zhiyi* in English word order. These three rules, along with some conventional phrase pairs, suffice to translate the sentence correctly:

[Aozhou] [shi] [[[yu [Beihan]$_1$ you [bangjiao]$_2$] de [shaoshu guojia]$_3$] zhiyi]

[Australia] [is] [one of [the [few countries]$_3$ that [have [dipl. rels.]$_2$ with [N. Korea]$_1$]]]

The system we describe in this article uses rules like (3), (4), and (5), which we formalize in the next section as rules of a **synchronous context-free grammar** (CFG).[1] Moreover, the system is able to learn them automatically from a parallel text without syntactic annotation.

Because our system uses a synchronous CFG, it could be thought of as an example of syntax-based statistical machine translation (MT), joining a line of research (Wu 1997; Alshawi, Bangalore, and Douglas 2000; Yamada and Knight 2001) that has been fruitful but has not previously produced systems that can compete with phrase-based systems in large-scale translation tasks such as the evaluations held by NIST. Our approach differs from early syntax-based statistical translation models in combining the idea of hierarchical structure with key insights from phrase-based MT: Crucially, by incorporating the use of elementary structures with possibly many words, we hope to inherit phrase-based MT's capacity for memorizing translations from parallel data. Other insights borrowed from the current state of the art include minimum-error-rate training of log-linear models (Och and Ney 2002; Och 2003) and use of an m-gram language model.

The conjunction of these various elements presents a considerable challenge for implementation, which we discuss in detail in this article. The result is the first system employing a grammar (to our knowledge) to perform better than phrase-based systems in large-scale evaluations.[2]

[1] The actual derivation used varies in practice. A previous version of the model selected precisely the derivation shown in the text, although the version described in this article happens to select a less intuitive one:

[Aozhou shi] [[[yu]$_1$ Beihan [you [bangjiao]$_2$ de [shaoshu]$_3$ guojia]] zhiyi .]

[Australia is] [one of the [[[few]$_3$ countries having [diplomatic relations]$_2$] [with]$_1$ North Korea] .]

[2] An earlier version of the system described in this article was entered by the University of Maryland as its primary system in the 2005 NIST MT Evaluation. The results can be found at `http://www.nist.gov/speech/tests/mt/mt05eval_official_results_release_20050801_v3.html`.

2. Related Work

Approaches to syntax-based statistical MT have varied in their reliance on syntactic theories, or annotations made according to syntactic theories. At one extreme are those, exemplified by that of Wu (1997), that have no dependence on syntactic theory beyond the idea that natural language is hierarchical. If these methods distinguish between different categories, they typically do not distinguish very many. Our approach, as presented here, falls squarely into this family. By contrast, other approaches, exemplified by that of Yamada and Knight (2001), do make use of parallel data with syntactic annotations, either in the form of phrase-structure trees or dependency trees (Ding and Palmer 2005; Quirk, Menezes, and Cherry 2005). Because syntactically annotated corpora are comparatively small, obtaining parsed parallel text in quantity usually entails running an automatic parser on a parallel corpus to produce noisy annotations.

Both of these strands of research have recently begun to explore extraction of larger rules, guided by word alignments. The extraction method we use, which is a straightforward generalization of phrase extraction from word-aligned parallel text, has been independently proposed before in various settings. The method of Block (2000) is the earliest instance we are aware of, though it is restricted to rules with one variable. The same method has also been used by Probst et al. (2002) and Xia and McCord (2004) in conjunction with syntactic annotations to extract rules that are used for reordering prior to translation. Finally, Galley et al. (2004) use the same method to extract a very large grammar from syntactically annotated data. The discontinuous phrases used by Simard et al. (2005) have a similar purpose to synchronous grammar rules; but they have variables that stand for single words rather than subderivations, and they can interleave in non-hierarchical ways.

3. Grammar

The model is based on a synchronous CFG, elsewhere known as a **syntax-directed transduction grammar** (Lewis and Stearns 1968). We give here an informal definition and then describe in detail how we build a synchronous CFG for our model.

3.1 Synchronous CFG

In a synchronous CFG the elementary structures are rewrite rules with aligned pairs of right-hand sides:

$$X \rightarrow \langle \gamma, \alpha, \sim \rangle$$

where X is a nonterminal, γ and α are both strings of terminals and nonterminals, and $\sim$ is a one-to-one correspondence between nonterminal occurrences in γ and nonterminal occurrences in α. For example, the hierarchical phrase pairs (3), (4), and (5) previously presented could be formalized in a synchronous CFG as:

$$\mathrm{X} \rightarrow \left\langle \text{yu } \mathrm{X}_{\boxed{1}} \text{ you } \mathrm{X}_{\boxed{2}}, \text{have } \mathrm{X}_{\boxed{2}} \text{ with } \mathrm{X}_{\boxed{1}} \right\rangle \tag{6}$$

$$\mathrm{X} \rightarrow \left\langle \mathrm{X}_{\boxed{1}} \text{ de } \mathrm{X}_{\boxed{2}}, \text{the } \mathrm{X}_{\boxed{2}} \text{ that } \mathrm{X}_{\boxed{1}} \right\rangle \tag{7}$$

$$X \rightarrow \left\langle X_{\boxed{1}} \text{ zhiyi}, \text{one of } X_{\boxed{1}} \right\rangle \tag{8}$$

where we have used boxed indices to indicate which nonterminal occurrences are linked by $\sim$. The conventional phrase pairs would be formalized as:

$$X \rightarrow \langle \text{Aozhou}, \text{Australia} \rangle \tag{9}$$

$$X \rightarrow \langle \text{Beihan}, \text{North Korea} \rangle \tag{10}$$

$$X \rightarrow \langle \text{shi}, \text{is} \rangle \tag{11}$$

$$X \rightarrow \langle \text{bangjiao}, \text{diplomatic relations} \rangle \tag{12}$$

$$X \rightarrow \langle \text{shaoshu guojia}, \text{few countries} \rangle \tag{13}$$

Two more rules complete our example:

$$S \rightarrow \left\langle S_{\boxed{1}} X_{\boxed{2}}, S_{\boxed{1}} X_{\boxed{2}} \right\rangle \tag{14}$$

$$S \rightarrow \left\langle X_{\boxed{1}}, X_{\boxed{1}} \right\rangle \tag{15}$$

A synchronous CFG derivation begins with a pair of linked start symbols. At each step, two linked nonterminals are rewritten using the two components of a single rule. When denoting links with boxed indices, we must consistently reindex the newly introduced symbols apart from the symbols already present. For an example using these rules, see Figure 1.

3.2 Rule Extraction

The bulk of the grammar consists of automatically extracted rules. The extraction process begins with a word-aligned corpus: a set of triples $\langle f, e, \sim \rangle$, where f is a French sentence, e is an English sentence, and $\sim$ is a (many-to-many) binary relation between positions of f and positions of e. The word alignments are obtained by running GIZA++ (Och and Ney 2000) on the corpus in both directions, and forming the union of the two sets of word alignments.

We then extract from each word-aligned sentence pair a set of rules that are consistent with the word alignments. This can be thought of in two steps. First, we identify **initial phrase** pairs using the same criterion as most phrase-based systems (Och and Ney 2004), namely, there must be at least one word inside one phrase aligned to a word inside the other, but no word inside one phrase can be aligned to a word outside the other phrase. For example, suppose our training data contained the fragment

30 多年来 的 友好 合作
30 duonianlai de youhao hezou
30 plus-years-past of friendly cooperation

friendly cooperation over the last 30 years

$$\langle \mathrm{S}_{\boxed{1}}, \mathrm{S}_{\boxed{1}} \rangle$$

$$\xRightarrow{(14)} \langle \mathrm{S}_{\boxed{2}} \mathrm{X}_{\boxed{3}}, \mathrm{S}_{\boxed{2}} \mathrm{X}_{\boxed{3}} \rangle$$

$$\xRightarrow{(14)} \langle \mathrm{S}_{\boxed{4}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}}, \mathrm{S}_{\boxed{4}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}} \rangle$$

$$\xRightarrow{(15)} \langle \mathrm{X}_{\boxed{6}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}}, \mathrm{X}_{\boxed{6}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}} \rangle$$

$$\xRightarrow{(9)} \langle \text{Aozhou } \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}}, \text{Australia } \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}} \rangle$$

$$\xRightarrow{(11)} \langle \text{Aozhou shi } \mathrm{X}_{\boxed{3}}, \text{Australia is } \mathrm{X}_{\boxed{3}} \rangle$$

$$\xRightarrow{(8)} \langle \text{Aozhou shi } \mathrm{X}_{\boxed{7}} \text{ zhiyi, Australia is one of } \mathrm{X}_{\boxed{7}} \rangle$$

$$\xRightarrow{(7)} \langle \text{Aozhou shi } \mathrm{X}_{\boxed{8}} \text{ de } \mathrm{X}_{\boxed{9}} \text{ zhiyi, Australia is one of the } \mathrm{X}_{\boxed{9}} \text{ that } \mathrm{X}_{\boxed{8}} \rangle$$

$$\xRightarrow{(6)} \langle \text{Aozhou shi yu } \mathrm{X}_{\boxed{1}} \text{ you } \mathrm{X}_{\boxed{2}} \text{ de } \mathrm{X}_{\boxed{9}} \text{ zhiyi,} \\ \text{Australia is one of the } \mathrm{X}_{\boxed{9}} \text{ that have } \mathrm{X}_{\boxed{2}} \text{ with } \mathrm{X}_{\boxed{1}} \rangle$$

$$\xRightarrow{(10)} \langle \text{Aozhou shi yu Beihan you } \mathrm{X}_{\boxed{2}} \text{ de } \mathrm{X}_{\boxed{9}} \text{ zhiyi,} \\ \text{Australia is one of the } \mathrm{X}_{\boxed{9}} \text{ that have } \mathrm{X}_{\boxed{2}} \text{ with North Korea} \rangle$$

$$\xRightarrow{(12)} \langle \text{Aozhou shi yu Beihan you bangjiao de } \mathrm{X}_{\boxed{9}} \text{ zhiyi,} \\ \text{Australia is one of the } \mathrm{X}_{\boxed{9}} \text{ that have diplomatic relations with North Korea} \rangle$$

$$\xRightarrow{(13)} \langle \text{Aozhou shi yu Beihan you bangjiao de shaoshu guojia zhiyi,} \\ \text{Australia is one of the few countries that have diplomatic relations with North Korea} \rangle$$

Figure 1
Example derivation of a synchronous CFG. Numbers above arrows are rules used at each step.

with word alignments as shown in Figure 2a. The initial phrases that would be extracted are shown in Figure 2b. More formally:

Definition 1
Given a word-aligned sentence pair $\langle f, e, \sim \rangle$, let f_i^j stand for the substring of f from position i to position j inclusive, and similarly for e_i^j. Then a rule $\langle f_i^j, e_{i'}^{j'} \rangle$ is an initial phrase pair of $\langle f, e, \sim \rangle$ iff:

1. $f_k \sim e_{k'}$ for some $k \in [i, j]$ and $k' \in [i', j']$;
2. $f_k \not\sim e_{k'}$ for all $k \in [i, j]$ and $k' \notin [i', j']$;
3. $f_k \not\sim e_{k'}$ for all $k \notin [i, j]$ and $k' \in [i', j']$.

Second, in order to obtain rules from the phrases, we look for phrases that contain other phrases and replace the subphrases with nonterminal symbols. For example, given the initial phrases shown in Figure 2b, we could form the rule

$$\mathrm{X} \rightarrow \langle \mathrm{X}_{\boxed{1}} \text{ duonianlai de } \mathrm{X}_{\boxed{2}}, \mathrm{X}_{\boxed{2}} \text{ over the last } \mathrm{X}_{\boxed{1}} \text{ years} \rangle \quad (16)$$

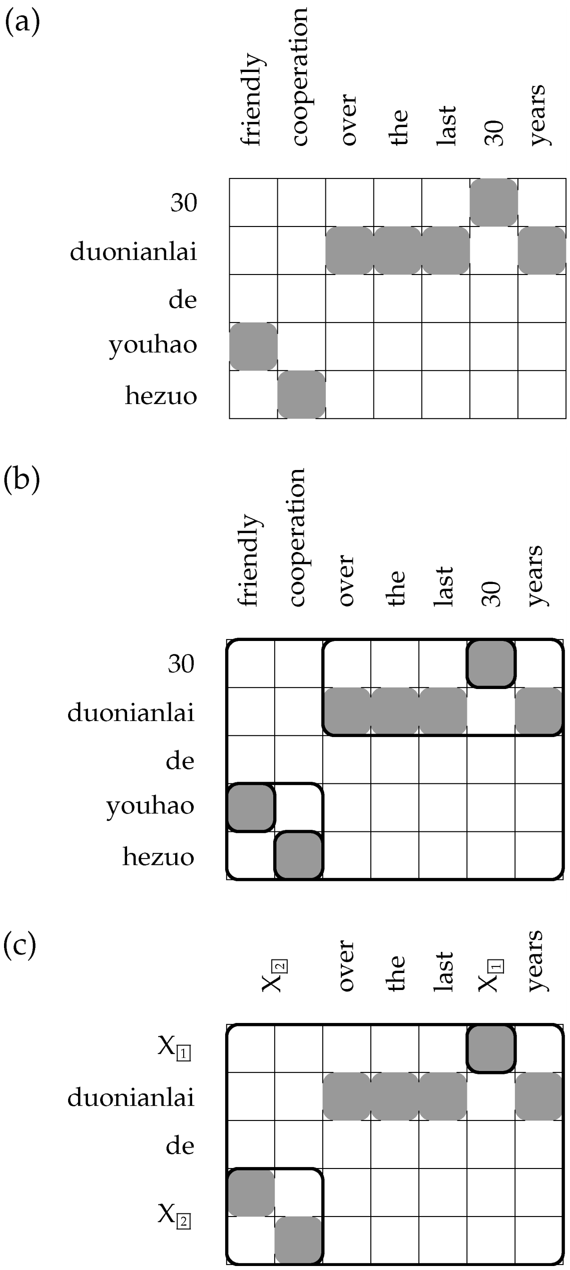


Figure 2
Grammar extraction example: (a) Input word alignment. (b) Initial phrases. (c) Example rule.

as shown in Figure 2c. More formally:

Definition 2
The set of rules of $\langle f, e, \sim\rangle$ is the smallest set satisfying the following:

1. If $\langle f_i^j, e_{i'}^{j'}\rangle$ is an initial phrase pair, then

$$X \rightarrow \langle f_i^j, e_{i'}^{j'}\rangle$$

is a rule of $\langle f, e, \sim\rangle$.

2. If $(X \rightarrow \langle \gamma, \alpha\rangle)$ is a rule of $\langle f, e, \sim\rangle$ and $\langle f_i^j, e_{i'}^{j'}\rangle$ is an initial phrase pair such that $\gamma = \gamma_1 f_i^j \gamma_2$ and $\alpha = \alpha_1 e_{i'}^{j'} \alpha_2$, then

$$X \rightarrow \left\langle \gamma_1 X_{\boxed{k}} \gamma_2, \alpha_1 X_{\boxed{k}} \alpha_2 \right\rangle$$

where k is an index not used in γ and α, is a rule of $\langle f, e, \sim\rangle$.

This scheme generates a very large number of rules, which is undesirable not only because it makes training and decoding very slow, but also because it creates **spurious ambiguity**—a situation where the decoder produces many derivations that are distinct yet have the same model feature vectors and give the same translation. This can result in *k*-best lists with very few different translations or feature vectors, which is problematic for the minimum-error-rate training algorithm (see Section 4.3). To avoid this, we filter our grammar according to the following constraints, chosen to balance grammar size and performance on our development set:

1. If there are multiple initial phrase pairs containing the same set of alignments, only the smallest is kept. That is, unaligned words are not allowed at the edges of phrases.
2. Initial phrases are limited to a length of 10 words on either side.
3. Rules are limited to five nonterminals plus terminals on the French side.
4. Rules can have at most two nonterminals, which simplifies the decoder implementation. This also makes our grammar weakly equivalent to an inversion transduction grammar (Wu 1997), although the conversion would create a very large number of new nonterminal symbols.
5. It is prohibited for nonterminals to be adjacent on the French side, a major cause of spurious ambiguity.
6. A rule must have at least one pair of aligned words, so that translation decisions are always based on some lexical evidence.

Variations of constraints (1) and (2) are also commonly used in phrase-based systems.

3.3 Other Rules

Glue rules. Having extracted rules from the training data, we could let X be the grammar's start symbol and translate new sentences using only the extracted rules. But for robustness and for continuity with phrase-based translation models, we allow the grammar to divide a French sentence into a sequence of chunks and translate one chunk at a time. We formalize this inside a synchronous CFG using the rules (14) and (15), which we call the **glue rules**, repeated here:

$$\mathrm{S} \rightarrow \left\langle \mathrm{S}_{\boxed{1}} \mathrm{X}_{\boxed{2}}, \mathrm{S}_{\boxed{1}} \mathrm{X}_{\boxed{2}} \right\rangle \tag{14}$$

$$\mathrm{S} \rightarrow \left\langle \mathrm{X}_{\boxed{1}}, \mathrm{X}_{\boxed{1}} \right\rangle \tag{15}$$

These rules analyze an S (the start symbol) as a sequence of Xs which are translated without reordering. Note that if we restricted our grammar to comprise only the glue rules and conventional phrase pairs (that is, rules without nonterminal symbols on the right-hand side), the model would reduce to a phrase-based model with monotone translation (no phrase reordering).

Entity rules. Finally, for each sentence to be translated, we run some specialized translation modules to translate the numbers, dates, numbers, and bylines in the sentence, and insert these translations into the grammar as new rules.[3] Such modules are often used by phrase-based systems as well, but here their translations can plug into hierarchical phrases, for example, into the rule

$$\mathrm{X} \rightarrow \left\langle \mathrm{X}_{\boxed{1}} \text{ duonianlai, over the last } \mathrm{X}_{\boxed{1}} \text{ years} \right\rangle \tag{17}$$

allowing it to generalize over numbers of years.

4. Model

Given a French sentence f, a synchronous CFG will have, in general, many derivations that yield f on the French side, and therefore (in general) many possible translations e. We now define a model over derivations D to predict which translations are more likely than others.

4.1 Definition

Following Och and Ney (2002), we depart from the traditional noisy-channel approach and use a more general log-linear model over derivations D:

$$P(D) \propto \prod_{i} \phi_i(D)^{\lambda_i} \tag{18}$$

[3] These modules are due to U. Germann and F. J. Och. In a previous paper (Chiang et al. 2005) we reported on translation modules for numbers and names. The present modules are not the same as those, though the mechanism for integrating them is identical.

where the ϕ_i are features defined on derivations and the λ_i are feature weights. One of the features is an m-gram language model $P_{LM}(e)$; the remainder of the features we will define as products of functions on the rules used in a derivation:

$$\phi_i(D) = \prod_{(X \rightarrow \langle \gamma, \alpha \rangle) \in D} \phi_i(X \rightarrow \langle \gamma, \alpha \rangle) \tag{19}$$

Thus we can rewrite $P(D)$ as

$$P(D) \propto P_{LM}(e)^{\lambda_{LM}} \times \prod_{i \neq LM} \prod_{(X \rightarrow \langle \gamma, \alpha \rangle) \in D} \phi_i(X \rightarrow \langle \gamma, \alpha \rangle)^{\lambda_i} \tag{20}$$

The factors other than the language model factor can be put into a particularly convenient form. A **weighted synchronous CFG** is a synchronous CFG together with a function w that assigns weights to rules. This function induces a weight function over derivations:

$$w(D) = \prod_{(X \rightarrow \langle \gamma, \alpha \rangle) \in D} w(X \rightarrow \langle \gamma, \alpha \rangle) \tag{21}$$

If we define

$$w(X \rightarrow \langle \gamma, \alpha \rangle) = \prod_{i \neq LM} \phi_i(X \rightarrow \langle \gamma, \alpha \rangle)^{\lambda_i} \tag{22}$$

then the probability model becomes

$$P(D) \propto P_{LM}(e)^{\lambda_{LM}} \times w(D) \tag{23}$$

It is easy to write dynamic-programming algorithms to find the highest-weight translation or k-best translations with a weighted synchronous CFG. Therefore it is problematic that $w(D)$ does not include the language model, which is extremely important for translation quality. We return to this challenge in Section 5.

4.2 Features

For our experiments, we use a feature set analogous to the default feature set of Pharaoh (Koehn, Och, and Marcu 2003). The rules extracted from the training bitext have the following features:

- $P(\gamma \mid \alpha)$ and $P(\alpha \mid \gamma)$, the latter of which is not found in the noisy-channel model, but has been previously found to be a helpful feature (Och and Ney 2002);

- the lexical weights $P_w(\gamma \mid \alpha)$ and $P_w(\alpha \mid \gamma)$, which estimate how well the words in α translate the words in γ (Koehn, Och, and Marcu 2003);[4]
- a penalty $\exp(-1)$ for extracted rules, analogous to Koehn's phrase penalty (Koehn 2003), which allows the model to learn a preference for longer or shorter derivations.

Next, there are penalties $\exp(-1)$ for various other classes of rules:

- for the glue rule (14), so that the model can learn a preference for hierarchical phrases over a serial combination of phrases,
- for each of the four types of rules (numbers, dates, names, bylines) inserted by the specialized translation modules, so that the model can learn how much to rely on each of them.

Finally, for all the rules, there is a word penalty $\exp(-\#T(\alpha))$, where $\#T$ just counts terminal symbols. This allows the model to learn a general preference for shorter or longer outputs.

4.3 Training

In order to estimate the parameters of the phrase translation and lexical-weighting features, we need counts for the extracted rules. For each sentence pair in the training data, there is in general more than one derivation of the sentence pair using the rules extracted from it. Because we have observed the sentence pair but have not observed the derivations, we do not know how many times each derivation has been seen, and therefore we do not actually know how many times each rule has been seen.

Following Och and others, we use heuristics to hypothesize a distribution of possible rules as though we observed them in the training data, a distribution that does not necessarily maximize the likelihood of the training data.[5] Och's method gives a count of one to each extracted phrase pair occurrence. We likewise give a count of one to each initial phrase pair occurrence, then distribute its weight equally among the rules obtained by subtracting subphrases from it. Treating this distribution as our observed data, we use relative-frequency estimation to obtain $P(\gamma \mid \alpha)$ and $P(\alpha \mid \gamma)$.

Finally, the parameters λ_i of the log-linear model (18) are learned by minimum-error-rate training (Och 2003), which tries to set the parameters so as to maximize the BLEU score (Papineni et al. 2002) of a development set. This gives a weighted synchronous CFG according to (22) that is ready to be used by the decoder.

[4] This feature uses word alignment information, which is discarded in the final grammar. If a rule occurs in training with more than one possible word alignment, Koehn, Och, and Marcu take the maximum lexical weight; we take a weighted average.

[5] This approach is similar to that taken by many parsers, such as SPATTER (Magerman 1995) and its successors, which use heuristics to hypothesize an augmented version of the training data, but it is especially reminiscent of the Data Oriented Parsing method (Bod 1992), which hypothesizes a distribution over many possible derivations of each training example from subtrees of varying sizes.

5. Decoding

In brief, our decoder is a CKY (Cocke-Kasami-Younger) parser with beam search together with a postprocessor for mapping French derivations to English derivations. Given a French sentence f, it finds the English yield of the single best derivation that has French yield f:

$$\hat{e} = e\left(\operatorname*{arg\,max}_{D \text{ s.t. } f(D) = f} P(D)\right) \tag{24}$$

Note that this is not necessarily the highest-probability English string, which would require a more expensive summation over derivations.

We now discuss the details of the decoder, focusing attention on efficiently calculating English language-model probabilities for possible translations, which is the primary technical challenge.

5.1 Basic Algorithm

In the following we present several parsers as deductive proof systems (Shieber, Schabes, and Pereira 1995; Goodman 1999). A parser in this notation defines a space of weighted **items**, in which some items are designated axioms and some items are designated **goals** (the items to be proven), and a set of inference rules of the form

$$\frac{I_1 : w_1 \quad \cdots \quad I_k : w_k}{I : w} \; \phi$$

which means that if all the items I_i (called the **antecedents**) are provable, with weight w_i, then I (called the **consequent**) is provable, with weight w, provided the side condition ϕ holds. The parsing process grows a set of provable items: It starts with the axioms, and proceeds by applying inference rules to prove more and more items until a goal is proven.

For example, the well-known CKY algorithm for CFGs in Chomsky normal form can be thought of as a deductive proof system whose items can take one of two forms:

- $[X, i, j]$, indicating that a subtree rooted in X has been recognized spanning from i to j (that is, spanning f_{i+1}^{j}), or
- $(X \rightarrow \gamma)$, if a rule $X \longrightarrow \gamma$ belongs to the grammar.[6]

The axioms would be

$$\frac{}{X \rightarrow \gamma : w} \quad (X \xrightarrow{w} \gamma) \in G$$

[6] Treating grammar rules as axioms is not standard practice, but advocated by Goodman (1999). Here, it has the benefit of simplifying the presentation in Section 5.3.4.

and the inference rules would be

$$\frac{Z \rightarrow f_{i+1} : w}{[Z, i, i+1] : w}$$

$$\frac{Z \rightarrow XY : w \quad [X, i, k] : w_1 \quad [Y, k, j] : w_2}{[Z, i, j] : w_1 w_2 w}$$

and the goal would be $[S, 0, n]$, where S is the start symbol of the grammar and n is the length of the input string f.

Given a synchronous CFG, we could convert its French-side grammar into Chomsky normal form, and then for each sentence, we could find the best parse using CKY. Then it would be a straightforward matter to revert the best parse from Chomsky normal form into the original form and map it into its corresponding English tree, whose yield is the output translation. However, because we have already restricted the number of nonterminal symbols in our rules to two, it is more convenient to use a modified CKY algorithm that operates on our grammar directly, without any conversion to Chomsky normal form. The axioms, inference rules, and goals for the basic decoder are shown in Figure 3. Its time complexity is $\mathcal{O}(n^3)$, just as CKY's is. Because this algorithm does not yet incorporate a language model, let us call it the $-$LM parser.

The actual search procedure is given by the pseudocode in Figure 4. It organizes the proved items into an array *chart* whose cells *chart*$[X, i, j]$ are sets of items. The cells are ordered such that every item comes after its possible antecedents: smaller spans before larger spans, and X items before S items (because of the unary rule $\mathrm{S} \rightarrow \langle \mathrm{X}_{\boxed{1}}, \mathrm{X}_{\boxed{1}} \rangle$). Then the parser can proceed by visiting the chart cells in order and trying to prove all the items for each cell. Whenever it proves a new item, it adds the item to the

$$\frac{}{X \rightarrow \gamma : w} \qquad (X \xrightarrow{w} \langle \gamma, \alpha \rangle) \in G$$

$$\frac{X \rightarrow f_{i+1}^{j} : w}{[X, i, j] : w}$$

$$\frac{Z \rightarrow f_{i+1}^{i_1} X f_{j_1+1}^{j} : w \quad [X, i_1, j_1] : w_1}{[Z, i, j] : w w_1}$$

$$\frac{Z \rightarrow f_{i+1}^{i_1} X f_{j_1+1}^{i_2} Y f_{j_2+1}^{j} : w \quad [X, i_1, j_1] : w_1 \quad [Y, i_2, j_2] : w_2}{[Z, i, j] : w w_1 w_2}$$

Goal item: $[S, 0, n]$

Figure 3
Inference rules for the $-$LM parser.

```
procedure PARSE
    for all axioms $(X \rightarrow \gamma)$ do
        add $(X \rightarrow \gamma)$ to rchart
    for $\ell \leftarrow 1 \ldots n$ do
        for all $i, j$ s.t. $j - i = \ell$ do
            if $\ell \leq \Lambda$ then
                for all items $[\mathrm{X}, i, j] : w$ inferable from items in rchart and chart do
                    add $[\mathrm{X}, i, j] : w$ to $chart[\mathrm{X}, i, j]$
            if $i = 0$ then
                for all items $[\mathrm{S}, i, j] : w$ inferable from items in rchart and chart do
                    add $[\mathrm{S}, i, j] : w$ to $chart[\mathrm{S}, i, j]$
```

Figure 4
Search procedure for the −LM parser.

appropriate chart cell; in order to reconstruct the derivations later, it must also store, with each item, a tuple of back-pointers to the antecedents from which the item was deduced (for axioms, an empty tuple is used). If two items are added to a cell that are equivalent except for their weights or back-pointers, then they are merged (in the MT decoding literature, this is also known as **hypothesis recombination**), with the merged item taking its weight and back-pointers from the better of the two equivalent items. (However, if we are interested in finding the *k*-best derivations, the merged item gets the multiset of all the tuples of back-pointers from the equivalent items. These back-pointers are used below in Section 5.2.)

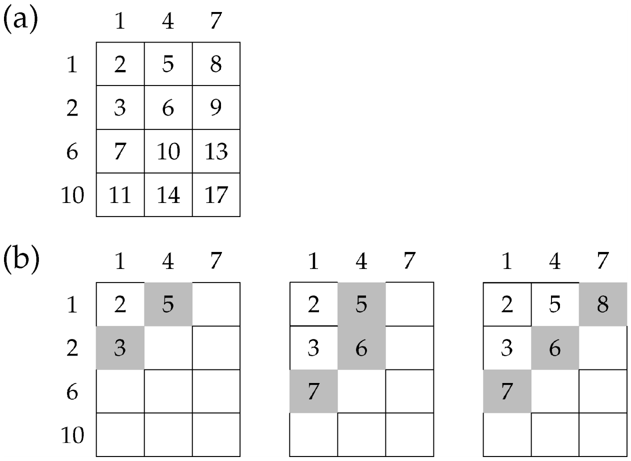
The algorithm in Figure 4 does not completely search the space of proofs, because it has a constraint that prohibits any X from spanning a substring longer than a fixed limit Λ on the French side, corresponding to the maximum length constraint on initial rules during training. This gives the decoding algorithm an asymptotic time complexity of $\mathcal{O}(n)$. In principle Λ should match the initial phrase length limit used in training (as it does in our experiments), but in practice it can be adjusted separately to maximize accuracy or speed.

5.2 Generating *k*-best Lists

We often want to find not only the best derivation for a French sentence but a list of the *k*-best derivations. These are used for minimum-error-rate training and for rescoring with a language model (Section 5.3.1). We describe here how to do this using the lazy algorithm of Huang and Chiang (2005). Part of this method will also be reused in our algorithm for fast parsing with a language model (Section 5.3.4).

If we conceive of lists as functions from indices to values, we may create a **virtual list,** a function that computes member values on demand instead of storing all the values statically. The heart of the *k*-best algorithm is a function MERGEPRODUCTS, which takes a set $\mathcal{L}$ of tuples of (virtual) lists with an operator $\otimes$ and returns a virtual list:

$$\textsc{MergeProducts}(\mathcal{L}, \otimes) = \bigcup_{\langle L_1, \ldots L_n \rangle \in \mathcal{L}} \left\{ \bigotimes_i x_i \;\middle|\; x_i \in L_i \right\} \tag{25}$$



(a)

	1	4	7
1	2	5	8
2	3	6	9
6	7	10	13
10	11	14	17

(b)

	1	4	7
1	2	5	
2	3		
6			
10			

1	4	7
2	5	
3	6	
7		

1	4	7
2	5	8
3	6	
7		

Figure 5
Example illustrating MERGEPRODUCTS, where $L_1 = \{1, 2, 6, 10\}$ and $L_2 = \{1, 4, 7\}$. Numbers are negative log-probabilities.

It assumes that the input lists are sorted and returns a sorted list. A naive implementation of MERGEPRODUCTS would simply calculate all possible products and sort; however, if we are only interested in the top part of the result, we can implement MERGEPRODUCTS so that the output values are computed lazily and the input lists are accessed only as needed. To do this, we must assume that the multiplication operator $\otimes$ is monotonic in each of its arguments. By way of motivation, consider the simple case $\mathcal{L} = \{\langle L_1, L_2\rangle\}$. The full set of possible products can be arranged in a two-dimensional grid (see Figure 5a), which we could then sort to obtain MERGEPRODUCTS($\mathcal{L}$). But because of our assumptions, we know that the first element of MERGEPRODUCTS($\mathcal{L}$) must be $L_1[1] \otimes L_2[1]$. Moreover, we know that the second element must be either $L_1[1] \otimes L_2[2]$ or $L_1[2] \otimes L_2[1]$. In general (see Figure 5b), if some of the cells have been previously enumerated, the next cell must be one of the cells (shaded gray) adjacent to the previously enumerated ones and we need not consider the others (shaded white). In this way, if we only want to compute the first few elements of MERGEPRODUCTS($\mathcal{L}$), we can do so by performing a small number of products and discarding the rest of the grid.

Figure 6 shows the pseudocode for MERGEPRODUCTS.[7] In lines 2–5, a priority queue is initialized with the best element from each $\mathbf{L} \in \mathcal{L}$, where $\mathbf{L}$ ranges over tuples of lists, and $\mathbf{1}$ stands for a vector whose elements all have the value 1 (the dimensionality of the vector should be evident from the context). The rest of the function creates the virtual list: To enumerate the next element of the list, we first insert the elements adjacent to the previously enumerated element, if any (lines 9–13, where $\mathbf{b}^i$ stands for the vector whose ith element is 1 and is zero elsewhere), and then enumerate the best element in the priority queue, if any (lines 14–18). We assume standard implementations of

[7] This version corrects the behavior of the previously published version in some boundary conditions. Thanks to D. Smith and J. May for pointing those cases out. In the actual implementation, an earlier version is used which has the correct behavior but not for cyclic forests (which the parser never produces).

```
function MERGEPRODUCTS($\mathcal{L}, \otimes$)
    $cand \leftarrow [], result \leftarrow []$
    for $\mathbf{L} \in \mathcal{L}$ do
        append $\langle \bigotimes_i L_i(1), \mathbf{L}, \mathbf{1} \rangle$ to $cand$
    HEAPIFY($cand$)
    $\mathbf{L} \leftarrow nil, \mathbf{r} \leftarrow nil$
    function LIST($n$)                    ▷ creates a closure including $cand, result, \mathbf{L}, \mathbf{r}$
        while $|result| < n$ do
            if $\mathbf{L} \neq nil$ then
                for $i \leftarrow 1 \ldots |\mathbf{L}|$ do
                    $\mathbf{r}' \leftarrow \mathbf{r} + \mathbf{b}^i$
                    if $L_i(r'_i)$ defined and $\langle \bigotimes_i L_i(r'_i), \mathbf{L}, \mathbf{r}' \rangle \notin cand$ then
                        INSERT($cand, \langle \bigotimes_i L_i(r'_i), \mathbf{L}, \mathbf{r}' \rangle$)
            if $|cand| > 0$ then
                $\langle x, \mathbf{L}, \mathbf{r} \rangle \leftarrow$ EXTRACTBEST($cand$)
                append $x$ to $result$
            else
                return undefined
        return $result[n]$
    return LIST
```

Figure 6
Function for computing the union of products of sorted lists (Huang and Chiang 2005).

the priority queue subroutines HEAPIFY, INSERT, and EXTRACTBEST (Cormen et al. 2001).

The *k*-best list generator is then easy to define (Figure 7). First, we generate a parse forest; then we simply apply MERGEPRODUCTS recursively to the whole forest, using memoization to ensure that we generate only one *k*-best list for each item in the forest. The pseudocode in Figure 7 will find only the weights for the *k*-best derivations; extending it to output the translations as well is a matter of modifying line 5 to package the English sides of rules together with the weights *w*, and replacing the real multiplication operator × in line 9 with one that not only multiplies weights but also builds partial translations out of subtranslations.

```
 1: function KBEST($v$)
 2:     if $best[v]$ not defined then
 3:         $\mathcal{L} \leftarrow \emptyset$
 4:         if $v$ an axiom then
 5:             $best[v] \leftarrow \{ w \mid v$ is an axiom with weight $w \}$, sorted best-first
 6:         else
 7:             for $\mathbf{u}$ s.t. $v$ was inferred from $\mathbf{u}$ do
 8:                 add $\langle$KBEST($u_1$), ..., KBEST($u_{|\mathbf{u}|}$)$\rangle$ to $\mathcal{L}$
 9:             $best[v] \leftarrow$ MERGEPRODUCTS($\mathcal{L}, \times$)
10:     return $best[v]$
```

Figure 7
Algorithm for computing *k*-best lists (Huang and Chiang 2005).

5.3 Adding the Language Model

We now turn to the problem of incorporating the language model (LM), describing three methods: first, using the $-$LM parser to obtain a k-best list of translations and rescoring it with the LM; second, incorporating the LM directly into the grammar in a construction reminiscent of the intersection of a CFG with a finite-state automaton; third, a hybrid method which we call **cube pruning**.

5.3.1 Rescoring. One easy way to incorporate the LM into the model would be to decode first using the $-$LM parser to produce a k-best list of translations, then to rescore the k-best list using the LM. This method has the potential to be very fast: linear in k. However, because the number of possible translations is exponential in n, we may have to set k extremely high in order to find the true best translation (taking the LM into account) or something acceptably close to it.

5.3.2 Intersection. A more principled solution would be to calculate the LM probabilities online. To do this, we view an m-gram LM as a weighted finite state machine M in which each state corresponds to a sequence of $(m-1)$ English terminal symbols. We can then intersect the English side of our weighted CFG G with this finite-state machine to produce a new weighted CFG that incorporates M. Thus P_{LM} would be part of the rule weights (22) just like the other features. (For notational consistency, however, we write the LM probabilities separately from the rule weights.) In principle this method should admit no search errors, though in practice the blow-up in the effective size of the grammar necessitates pruning of the search space, which can cause search errors.

The classic construction for intersecting a (non-synchronous) CFG with a finite-state machine is due to Bar-Hillel, Perles, and Shamir (1964), but we use a slightly different construction proposed by Wu (1996) for inversion transduction grammar and bigram LMs. We present an adaptation of his algorithm to synchronous CFGs with two nonterminals per right-hand side and general m-gram LMs. First, assume that the LM expects a whole sentence to be preceded by $(m-1)$ start-of-sentence symbols $\langle s\rangle$ and followed by a single end-of-sentence symbol $\langle /s\rangle$. The grammar can be made to do this simply by adding a rule

$$\mathrm{S}' \rightarrow \langle \mathrm{S}_{\boxed{1}}, \langle s\rangle^{m-1}\, \mathrm{S}_{\boxed{1}}\ \langle /s\rangle\rangle \tag{26}$$

and making S′ the new start symbol.

First, we define two functions p and q which operate on strings over $T \cup \{\star\}$, where T is the English terminal alphabet, and $\star$ is a special placeholder symbol that stands for an elided part of an English string.

$$p(a_1 \cdots a_\ell) = \prod_{\substack{m \leq i \leq \ell \\ \star \notin \{a_{i-m+1}, \ldots, a_i\}}} P_{LM}(a_i \mid a_{i-m+1} \cdots a_{i-1}) \tag{27}$$

$$q(a_1 \cdots a_\ell) = \begin{cases} a_1 \cdots a_{m-1} \star a_{\ell-m+2} \cdots a_\ell & \text{if } \ell \geq m \\ a_1 \cdots a_\ell & \text{otherwise} \end{cases} \tag{28}$$

Table 1
Values of p and q in the "cgisf" example.

$a_1 \cdots a_\ell$	$p(a_1 \cdots a_\ell)$	$q(a_1 \cdots a_\ell)$
c g i	$P_{LM}(\text{i} \mid \text{c g})$	c g $\star$ g i
s f	1	s f
c g $\star$ g i s f	$P_{LM}(\text{s} \mid \text{g i}) \times P_{LM}(\text{f} \mid \text{i s})$	c g $\star$ s f
$\langle s \rangle$ $\langle s \rangle$ c g $\star$ s f $\langle /s \rangle$	$P_{LM}(\text{c} \mid \langle s \rangle\, \langle s \rangle) \times P_{LM}(\text{g} \mid \langle s \rangle\, \text{c}) \times P_{LM}(\langle /s \rangle \mid \text{s f})$	$\langle s \rangle$ $\langle s \rangle$ $\star$ f $\langle /s \rangle$

The function p calculates LM probabilities for all the complete m-grams in a string; the function q elides symbols when all their m-grams have been accounted for. These functions let us correctly calculate the LM score of a sentence piecemeal. For example, let $m = 3$ and "c g i s f" stand for "colorless green ideas sleep furiously." Then Table 1 shows some values of p and q.

Then we may extend the $-$LM parser as shown in Figure 8 to use p and q to calculate LM probabilities. We call this parser the +LM parser. The items are of the form $[X, i, j; e]$, signifying that a subtree rooted in X has been recognized spanning from i to j on the French side, and its English translation (possibly with parts elided) is e.

The theoretical running time of this algorithm is $\mathcal{O}(n^3|T|^{4(m-1)})$, because a deduction can combine up to two starred strings, which each have up to $2(m-1)$ terminal symbols. This is far too slow to use in practice, so we must use beam-search to prune the search space down to a reasonable size.

5.3.3 Pruning. The chart is organized into cells, each of which contains all the items standing for X spanning f_{i+1}^{j}. The rule items are also organized into cells, each of which contains all the rules with the same French side and left-hand side. From here on, let us

$$\frac{}{X \to \langle \gamma, \alpha \rangle : w} \qquad (X \xrightarrow{w} \langle \gamma, \alpha \rangle) \in G$$

$$\frac{X \to \langle f_{i+1}^{j}, \alpha \rangle : w}{[X, i, j; q(\alpha)] : wp(\alpha)}$$

$$\frac{Z \to \langle f_{i+1}^{i_1} X f_{j_1+1}^{j}, \alpha \rangle : w \quad [X, i_1, j_1; e_1] : w_1}{[Z, i, j; q(\alpha')] : ww_1p(\alpha')} \qquad \alpha' = \alpha[e_1/X]$$

$$\frac{Z \to \langle f_{i+1}^{i_1} X_{\boxed{1}} f_{j_1+1}^{i_2} Y_{\boxed{2}} f_{j_2+1}^{j}, \alpha \rangle : w \quad [X, i_1, j_1; e_1] : w_1 \quad [Y, i_2, j_2; e_2] : w_2}{[Z, i, j; q(\alpha')] : ww_1w_2p(\alpha')} \qquad \alpha' = \alpha[e_1/X_{\boxed{1}}, e_2/Y_{\boxed{2}}]$$

$$\text{Goal item: } [\text{S}, 0, n; \langle s \rangle^{m-1} \star e \langle /s \rangle]$$

Figure 8
Inference rules for the +LM parser. Here $w[x/X]$ means the string w with the string x substituted for the symbol X. The function q is defined in the text.

consider the item scores as costs, that is, negative log (base-10) probabilities. Then, for each cell, we throw out any item that has a score worse than:

- β plus the best score in the same cell, or
- the score of the bth best item in the same cell.

In the +LM parser, the score of an item $[X, i, j; e]$ in the chart does not reflect the LM probability of generating the first $(m-1)$ words of e. Thus two items $[X, i, j; e]$ and $[X, i, j; e']$ are not directly comparable. To enable more meaningful comparisons, we define a heuristic

$$h([X, i, j; e]) = -\log_{10} P_{LM}(e_1 \cdots e_\ell) \qquad \text{where } \ell = \min\{m-1, |e|\} \tag{29}$$

Similarly for rules,

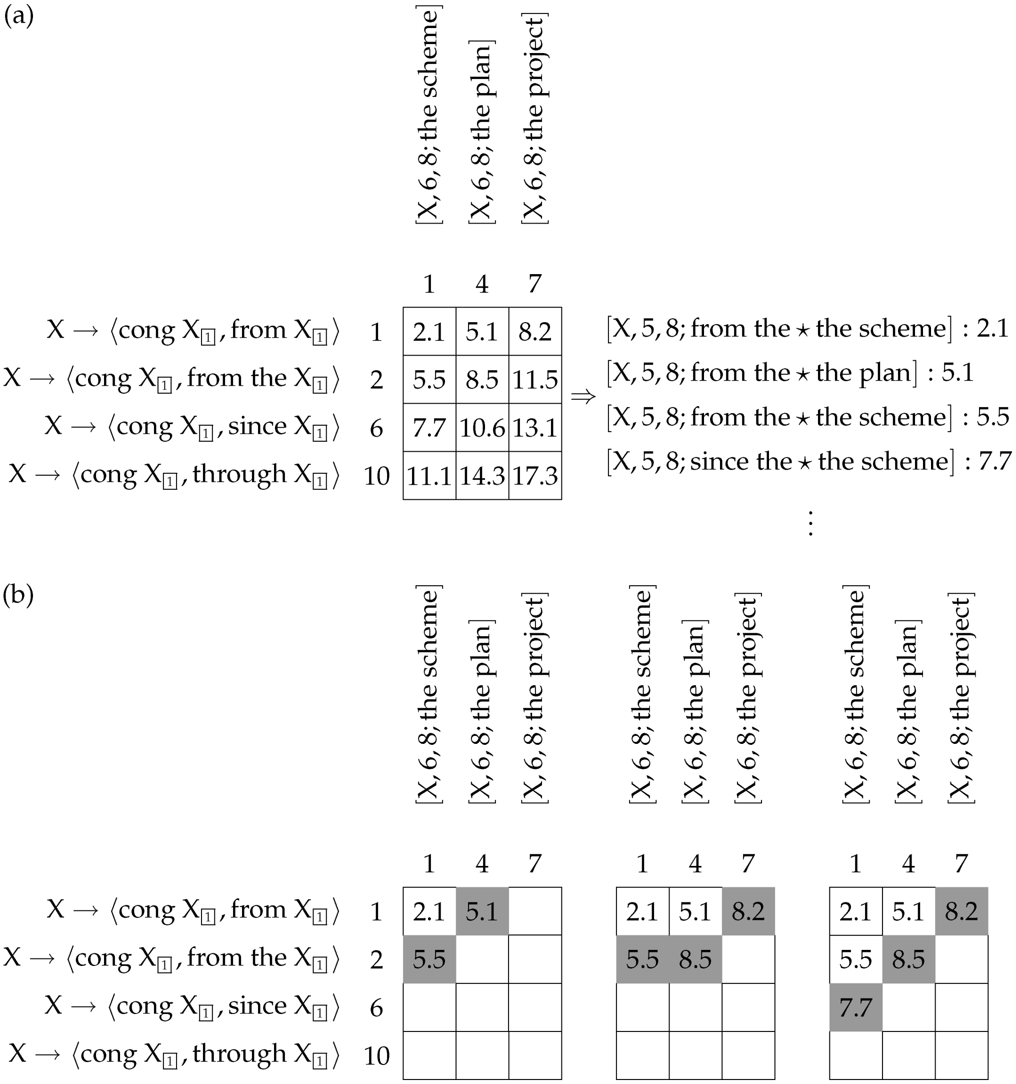
$$h(X \rightarrow \langle \gamma, w_0 X_1 w_1 \cdots w_{r-1} X_r w_r \rangle) = -\log_{10} \prod_{i=0}^{r} P_{LM}(w_i) \tag{30}$$

When comparing items for pruning (and only for pruning), we add this heuristic function to the score of each item.

5.3.4 Cube Pruning. Now we can develop a compromise between the rescoring and intersection methods. Consider Figure 9a. To the left of the grid we have four rules with the same French side, and above we have three items with the same category and span, that is, they belong to the same chart cell. Any of the twelve combinations of these rules and items can be used to deduce a new item (whose scores are shown in the grid), and all these new items will go into the same chart cell (partially listed on the right). The intersection method would compute all twelve items and add them to the new chart cell, where most of them will likely be pruned away. In actuality, the grid may be a cube (one dimension for rules and two dimensions for two nonterminals) with up to b^3 elements, whereas the target chart cell can hold at most b items (where b is the limit on the size of the cell imposed during pruning). Thus the vast majority of computed items are pruned. But it is possible to compute only a small corner of the cube and preemptively prune the rest of the items without computing them, a method we refer to as **cube pruning**.

The situation pictured in Figure 9a is very similar to k-best list generation. The four rules to the left of the grid can be thought of like a 4-best list for a single −LM rule item (X → cong X); the three items above the grid, like a 3-best list for the single −LM item [X, 6, 8]; and the new items to be deduced, like a k-best list for [X, 5, 8], except that we don't know what k is in advance. If we could use MERGEPRODUCTS to enumerate the new items best-first, then we could enumerate them until one of them was pruned from the new cell; then the rest of items, which would have a worse score than the pruned item, could be preemptively pruned.

MERGEPRODUCTS expects its input lists to be sorted best-first, and the $\otimes$ operator to be monotonic in each of its arguments. For cube pruning, we sort items (both in the inputs to MERGEPRODUCTS and in the priority queue inside MERGEPRODUCTS) according to their +LM score, including the heuristic function h. The $\otimes$ operator we use takes one or more antecedent items and forms their consequent item according to



(a)

		[X, 6, 8; the scheme]	[X, 6, 8; the plan]	[X, 6, 8; the project]
		1	4	7
$X \rightarrow \langle \text{cong } X_{\boxed{1}}, \text{from } X_{\boxed{1}} \rangle$	1	2.1	5.1	8.2
$X \rightarrow \langle \text{cong } X_{\boxed{1}}, \text{from the } X_{\boxed{1}} \rangle$	2	5.5	8.5	11.5
$X \rightarrow \langle \text{cong } X_{\boxed{1}}, \text{since } X_{\boxed{1}} \rangle$	6	7.7	10.6	13.1
$X \rightarrow \langle \text{cong } X_{\boxed{1}}, \text{through } X_{\boxed{1}} \rangle$	10	11.1	14.3	17.3

⇒

[X, 5, 8; from the ⋆ the scheme] : 2.1
[X, 5, 8; from the ⋆ the plan] : 5.1
[X, 5, 8; from the ⋆ the scheme] : 5.5
[X, 5, 8; since the ⋆ the scheme] : 7.7
⋮

(b)

		[X, 6, 8; the scheme]	[X, 6, 8; the plan]	[X, 6, 8; the project]
		1	4	7
$X \rightarrow \langle \text{cong } X_{\boxed{1}}, \text{from } X_{\boxed{1}} \rangle$	1	2.1	5.1	
$X \rightarrow \langle \text{cong } X_{\boxed{1}}, \text{from the } X_{\boxed{1}} \rangle$	2	5.5		
$X \rightarrow \langle \text{cong } X_{\boxed{1}}, \text{since } X_{\boxed{1}} \rangle$	6			
$X \rightarrow \langle \text{cong } X_{\boxed{1}}, \text{through } X_{\boxed{1}} \rangle$	10			

[X, 6, 8; the scheme]	[X, 6, 8; the plan]	[X, 6, 8; the project]
1	4	7
2.1	5.1	8.2
5.5	8.5	

[X, 6, 8; the scheme]	[X, 6, 8; the plan]	[X, 6, 8; the project]
1	4	7
2.1	5.1	8.2
5.5	8.5	
7.7		

Figure 9
Example illustrating hybrid method for incorporating the LM. Numbers are negative log-probabilities. (a) *k*-best list generation. (b) Cube pruning.

the +LM parser. Note that the LM makes this ⊗ only approximately monotonic. This means that the enumeration of new items will not necessarily be best-first. To alleviate this problem, we stop the enumeration not as soon as an item falls outside the beam, but as soon as an item falls outside the beam by a margin of ε. This quantity ε expresses our guess as to how much the scores of the enumerated items can fluctuate because of the LM. A simpler approach, and probably better in practice, would be simply to set $\varepsilon = 0$, that is, to ignore any fluctuation, but increase β and b to compensate.

See Figure 9b for an example of cube pruning. The upper-left grid cell is enumerated first, as in the *k*-best example in Section 5.2, but the choice of the second is different, because of the added LM costs. Then, the third item is enumerated and merged with the first (unlike in the *k*-best algorithm). Supposing a threshold beam of $\beta = 5$ and a margin of $\varepsilon = 0.5$, we quit upon considering the next item, because, with a score of 7.7, it falls outside the beam by more than ε. The rest of the grid is then discarded.

The pseudocode is given in Figure 10. The function INFER+LM is used as the $\otimes$ operator; it takes a tuple of antecedent +LM items and returns a consequent +LM item according to the inference rules in Figure 8. The procedure REPARSE+LM takes a −LM chart *chart* as input and produces a +LM chart *chart*′. The variables u, v stand for items in −LM and u', v', for items in +LM, and the relation $v \sqsupset v'$ is defined as follows:

$$[X, i, j] \sqsupset [X, i, j; e] \tag{31}$$

$$(X \rightarrow \gamma) \sqsupset (X \rightarrow \langle \gamma, \alpha \rangle) \tag{32}$$

For each cell in the input chart, it takes the single item from the cell and constructs the virtual list L of all of its +LM counterparts (lines 9–15). Then, it adds the top items of L to the target cell until the cell is judged to be full (lines 16–20).

6. Experiments

The implementation of our system, named Hiero, is in Python, a bytecode-interpreted language, and optimized using Psyco, a just-in-time compiler (Rigo 2004), and Pyrex, a Python-like compiled language, with C++ code from the SRI Language Modeling Toolkit (Stolcke 2002). In this section we report on experiments with Mandarin-to-English translation. Our evaluation metric is case-insensitive BLEU-4 (Papineni et al. 2002), as defined by NIST, that is, using the shortest (as opposed to closest) reference sentence length for the brevity penalty.

```
 1: function INFER+LM($\mathbf{u}'$)
 2:     return the unique $v'$ inferable from $\mathbf{u}'$ in +LM
 3:
 4: procedure REPARSE+LM
 5:     for $\ell \leftarrow 1 \dots n$ do
 6:         for all $i, j$ s.t. $j - i = \ell$ do
 7:             for all $A = \mathrm{X}, \mathrm{S}$ do
 8:                 $v \leftarrow chart[A, i, j]$
 9:                 if $v$ an axiom then
10:                     $L \leftarrow \{ v' \mid v \sqsupset v' \}$, sorted best-first
11:                 else
12:                     $\mathcal{L} \leftarrow \emptyset$
13:                     for $\mathbf{u}$ s.t. $v$ was inferred from $\mathbf{u}$ do
14:                         add $\langle chart[u_1], \dots, chart[u_{|\mathbf{u}|}] \rangle$ to $\mathcal{L}$
15:                     $L \leftarrow$ MERGEPRODUCTS($\mathcal{L}$, INFER+LM)
16:                 $i \leftarrow 0$
17:                 repeat
18:                     $i \leftarrow i + 1$
19:                     add $L[i]$ to $chart'[v]$
20:                 until $L[i]$ falls outside beam by a margin of $\varepsilon$
```

Figure 10
Algorithm for faster integrated calculation of LM probabilities.

6.1 Experimental Setup

We ran the grammar extractor of Section 3.2 on the parallel corpora listed in Table 2 with the exception of the United Nations data, for a total of 28 million words (English side).[8] We then filtered this grammar for our development set, which was the 2002 NIST MT evaluation dry-run data, and our test sets, which were the data from the 2003–2005 NIST MT evaluations. Some example rules are shown in Table 3, and the sizes of the filtered grammars are shown in Table 4.

We also used the SRI Language Modeling Toolkit to train two trigram language models with modified Kneser–Ney smoothing (Kneser and Ney 1995; Chen and Goodman 1998): one on 2.8 billion words from the English Gigaword corpus, and the other on the English side of the parallel text (28 million words).

6.2 Evaluating Translation Speed

Table 5 shows the average decoding time on part of the development set for the three LM-incorporation methods described in Section 5.3, on a single processor of a dual 3 GHz Xeon machine. For these experiments, only the Gigaword language model was used. We set $b = 30, \beta = 1$ for X cells, $b = 15, \beta = 1$ for S cells, and $b = 100$ for rules except where noted in Table 5. Note that values for β and ε are only meaningful relative to the scale of the feature weights; here, the language model weight was 0.06. The feature weights were obtained by minimum-error-rate training using the cube-pruning ($\varepsilon = 0.1$) decoder. For the LM rescoring decoder, parsing and k-best list generation used feature weights optimized for the $-$LM model, but rescoring used the same weights as the other experiments.

We tested the rescoring method ($k = 10^3$ and 10^4), the intersection method, and the cube-pruning method ($\varepsilon = 0$, 0.1, and 0.2). The LM rescoring decoder ($k = 10^4$) is the fastest but has the poorest BLEU score. Identifying and rescoring the k-best derivations is very quick; the execution time is dominated by reconstructing the output strings for the k-best derivations, so it is possible that further optimization could reduce these times. The intersecting decoder has the best score but runs very slowly. Finally, the cube-pruning decoder runs almost as fast as the rescoring decoder and translates almost as well as the intersecting decoder. Among these tests, $\varepsilon = 0.1$ gives the best results, but in general the optimal setting will depend on the other beam settings and the scale of the feature weights.

6.3 Evaluating Translation Accuracy

We compared Hiero against two baselines: the state-of-the-art phrase-based system ATS (Och et al. 2004; Thayer et al. 2004), and Hiero itself run as a conventional phrase-based system with monotone translation (no phrase reordering).

The ATS baseline was trained on all the parallel data listed in Table 1, for a total of 159 million words (English side). The second language model was also trained on the English side of the whole bitext. Phrases of up to 10 in length on the French side were extracted from the parallel text, and minimum-error-rate training (Och 2003) was

[8] We can train on the full training data shown if tighter constraints are placed on rule extraction for the United Nations data. For example, extracting only rules with no variables up to length 5 yields a grammar of 5.8M rules (filtered for the development set), which fits into memory easily.

Table 2
Corpora used in training data. Sizes are approximate and in millions of words.

Corpus	LDC catalog	Size
United Nations	LDC2004E12	112
Hong Kong Hansards	LDC2004T08	12
FBIS	LDC2003E14	10
Xinhua	LDC2002E18	4
Sinorama	LDC2005E47	3
Named entity list	LDC2003E01	1
Multiple Translation Chinese	LDC2002T01, LDC2003T17, LDC2004T07	0.8
Chinese Treebank	LDC2002E17, LDC2003E07	0.2
Translation lexicon	LDC2002L27	0.1
Chinese News Translation	LDC2005T06	0.1
English Gigaword (2nd ed.)	LDC2005T12	2800

performed on the development set for 17 features, the same as used in the NIST 2004 and 2005 evaluations.[9] These features are similar to the features used for our system, but also include features for phrase-reordering (which are not applicable to our system), IBM Model 1 in both directions, a missing word penalty, and a feature that controls a fallback lexicon.

The other baseline, which we call Hiero Monotone, is the same as Hiero except with the limitation that extracted rules cannot have any nonterminal symbols on their right-hand sides. In other words, only conventional phrases can be extracted, of length up to 5. These phrases are combined using the glue rules only, which makes the grammar equivalent to a conventional phrase-based model with monotone translation. Thus this system represents the nearest phrase-based equivalent to our model, to provide a controlled test of the effect of hierarchical phrases.

We performed minimum-error-rate training separately on Hiero and Hiero Monotone to maximize their BLEU scores on the development set; the feature weights for Hiero are shown in Table 6. The beam settings used for both decoders were $\beta = 30, b = 30$ for X cells, $\beta = 30, b = 15$ for S cells, $b = 100$ for rules, and $\varepsilon = 3$. On the test set, we found that Hiero improves over both baselines in all three tests (see Table 7). All improvements are statistically significant ($p < 0.01$) using the sign test as described by Collins, Koehn, and Kučerová (2005).

7. Conclusion

Syntax-based statistical machine translation is a twofold challenge. It is a modeling challenge, in part because of the difficulty of coordinating syntactic structures with potentially messy parallel corpora; it is an implementation challenge, because of the added complexity introduced by hierarchical structures. Here we have addressed the modeling challenge by taking only the fundamental idea from syntax, that language is hierarchically structured, and integrating it conservatively into a phrase-based model typical of the current state of the art. This fusion does no violence to the latter; indeed, we have presented our approach as a logical outgrowth of the phrase-based approach. Moreover, hierarchical structure improves translation accuracy significantly.

[9] The definition of BLEU used in this training was the original IBM definition (Papineni et al. 2002), which defines the effective reference length as the reference length that is closest to the test sentence length.

Table 3
A selection of extracted rules, with ranks after filtering for the development set. All have X for their left-hand sides.

Rank	Chinese	English
1	。	.
2	，	,
3	de	the
4	,	,
5	he	and
6	$X_{\boxed{1}}$.	$X_{\boxed{1}}$.
7	、	,
8	zai	in
9	， $X_{\boxed{1}}$	, $X_{\boxed{1}}$
10	$X_{\boxed{1}}$ ，	$X_{\boxed{1}}$,
11	de	of
12	，	.
13	“	"
14	”	"
15	、	and
63	zai $X_{\boxed{1}}$	in $X_{\boxed{1}}$
394	$X_{\boxed{1}}$ de $X_{\boxed{2}}$	the $X_{\boxed{2}}$ of $X_{\boxed{1}}$
756	$X_{\boxed{1}}$ de $X_{\boxed{2}}$	the $X_{\boxed{2}}$ $X_{\boxed{1}}$
1061	$X_{\boxed{1}}$ de $X_{\boxed{2}}$	the $X_{\boxed{2}}$ of the $X_{\boxed{1}}$
3752	$X_{\boxed{1}}$ hou	after $X_{\boxed{1}}$
4399	jingtian $X_{\boxed{1}}$	$X_{\boxed{1}}$ this year
5232	$X_{\boxed{1}}$ yuan	\$ $X_{\boxed{1}}$
5506	zhongguo $X_{\boxed{1}}$	$X_{\boxed{1}}$ of china
6030	$X_{\boxed{1}}$ duo	more than $X_{\boxed{1}}$
7947	$X_{\boxed{1}}$ zhongguo $X_{\boxed{2}}$	$X_{\boxed{2}}$ $X_{\boxed{1}}$ china
8119	$X_{\boxed{1}}$ zai $X_{\boxed{2}}$	in $X_{\boxed{2}}$ $X_{\boxed{1}}$
11052	zai $X_{\boxed{1}}$ zhong $X_{\boxed{2}}$	in $X_{\boxed{1}}$ $X_{\boxed{2}}$
11996	zai $X_{\boxed{1}}$ wenti shang	on the $X_{\boxed{1}}$ issue
12068	zai $X_{\boxed{1}}$ de wenti shang	on the basis of $X_{\boxed{1}}$
13237	zai $X_{\boxed{1}}$ hou	after $X_{\boxed{1}}$
14125	zai $X_{\boxed{1}}$ zhiqian	before $X_{\boxed{1}}$
14249	$X_{\boxed{1}}$ nian $X_{\boxed{2}}$	$X_{\boxed{2}}$ $X_{\boxed{1}}$ years
16871	zai $X_{\boxed{1}}$ de $X_{\boxed{2}}$	$X_{\boxed{2}}$ in $X_{\boxed{1}}$
28145	$X_{\boxed{1}}$ guanxi de $X_{\boxed{2}}$	the $X_{\boxed{2}}$ of $X_{\boxed{1}}$ relations
34294	zai $X_{\boxed{1}}$ de $X_{\boxed{2}}$ xia	under the $X_{\boxed{2}}$ of $X_{\boxed{1}}$

Table 4
Test set sizes, with grammar sizes for two systems.

Test set	Sentences	Phrases/rules (thousands)	
		Hiero Monotone	Hiero
development	993	448	3712
MT03	919	417	3389
MT04	1788	643	5556
MT05	1082	455	3646

Table 5
Comparison of three methods for decoding with a language model. Time = mean per-sentence user+system time, in seconds. BLEU = case-insensitive BLEU-4. All tests were on the first 400 sentences of the development set.

Method	Settings	Time	BLEU
rescore	$k = 10^4$	16	33.31
rescore	$k = 10^5$	139	33.33
intersect*		1455	37.09
cube prune	$\varepsilon = 0$	23	36.14
cube prune	$\varepsilon = 0.1$	35	36.77
cube prune	$\varepsilon = 0.2$	111	36.91

*Rules were pruned using $b = 30$, $\beta = 1$.

Table 6
Feature weights obtained by minimum-error-rate training.

Feature	Weight
language model (large)	1.00
language model (bitext)	1.03
$P(\gamma \mid \alpha)$	0.155
$P(\alpha \mid \gamma)$	1.23
$P_w(\gamma \mid \alpha)$	1.61
$P_w(\alpha \mid \gamma)$	0.494
numbers	0.364
dates	6.67
names	2.89
bylines	−952
extracted rules	4.32
glue rule	−0.281
word penalty	−4.12

The choice to use hierarchical structures that are more complex than flat structures, as well as rules that contain multiple lexical items instead of one, an m-gram model whose structure cuts across the structure of context-free derivations, and large amounts of training data for meaningful comparison with modern systems—these all threaten to make training a synchronous grammar and translating with it intractable. We have shown how, through training with simple methods inspired by phrase-based models, and translating using a modified CKY with cube pruning, this challenge can be met.

Table 7
Results on baseline systems and hierarchical system. Also shown are the 95% confidence intervals, obtained using bootstrap resampling.

System	MT03	MT04	MT05
Hiero Monotone	28.27 ± 1.03	28.83 ± 0.74	26.35 ± 0.92
ATS	30.84 ± 0.99	31.74 ± 0.73	30.50 ± 0.95
Hiero	33.72 ± 1.12	34.57 ± 0.82	31.79 ± 0.91

Clearly, however, we have only scratched the surface of the modeling challenge. The fact that moving from flat structures to hierarchical structures significantly improves translation quality suggests that more specific ideas from syntax may be valuable as well. There are many possibilities for enriching the simple framework that the present model provides. But the course taken here is one of organic development of an approach known to work well at large-scale tasks, and we plan to stay this course in future work towards more syntactically informed statistical machine translation.

Acknowledgments

I would like to thank Liang Huang, Philipp Koehn, Adam Lopez, Nitin Madnani, Daniel Marcu, Christof Monz, Dragos Munteanu, Philip Resnik, Michael Subotin, Wei Wang, and the anonymous reviewers. This work was partially supported by ONR MURI contract FCPO.810548265, by Department of Defense contract RD-02-5700, and under the GALE program of the Defense Advanced Research Projects Agency, contract HR 0011-06-C-0022. *S. D. G.*

References

Alshawi, Hiyan, Srinivas Bangalore, and Shona Douglas. 2000. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26:45–60.

Bar-Hillel, Yehoshua, M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In Yehoshua Bar-Hillel, editor, *Language and Information: Selected Essays on their Theory and Application*. Addison-Wesley, Reading, MA, pages 116–150.

Block, Hans Ulrich. 2000. Example-based incremental synchronous interpretation. In Wolfgang Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer-Verlag, Berlin, pages 411–417.

Bod, Rens. 1992. A computational model of language performance: Data Oriented Parsing. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING)*, pages 855–859, Nantes, France.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Chen, Stanley F. and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.

Chiang, David. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 263–270, Ann Arbor, MI.

Chiang, David, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. 2005. The Hiero machine translation system: Extensions, evaluation, and analysis. In *Proceedings of HLT/EMNLP 2005*, pages 779–786, Vancouver, Canada.

Collins, Michael, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 531–540, Ann Arbor, MI.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms*. MIT Press, second edition, Cambridge, MA.

Ding, Yuan and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 541–548, Ann Arbor, MI.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL 2004*, pages 273–280, Boston, MA.

Goodman, Joshua. 1999. Semiring parsing. *Computational Linguistics*, 25:573–605.

Huang, Liang and David Chiang. 2005. Better *k*-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT)*, pages 53–64, Vancouver, Canada.

Kneser, Reinhard and Hermann Ney. 1995. Improved backing-off for *M*-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184, Detroit, MI.

Koehn, Philipp. 2003. *Noun Phrase Translation*. Ph.D. thesis, University of Southern California, Los Angeles, CA.

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

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

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

Magerman, David M. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the ACL*, pages 276–283, Cambridge, MA.

Marcu, Daniel and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP 2002*, pages 133–139, Philadelphia, PA.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 160–167, Sapporo, Japan.

Och, Franz Josef and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the ACL*, pages 440–447, Hong Kong.

Och, Franz Josef and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 295–302, Philadelphia, PA.

Och, Franz Josef and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449.

Och, Franz Josef, Ignacio Thayer, Daniel Marcu, Kevin Knight, Dragos Stefan Munteanu, Quamrul Tipu, Michel Galley, and Mark Hopkins. 2004. Arabic and Chinese MT at USC/ISI. Presentation given at NIST Machine Translation Evaluation Workshop, Alexandria, VA.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 311–318, Philadelphia, PA.

Probst, Katharina, Lori Levin, Erik Peterson, Alon Lavie, and Jaime Carbonell. 2002. MT for minority languages using elicitation-based learning of syntactic transfer rules. *Machine Translation*, 17:245–270.

Quirk, Chris, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 271–279, Ann Arbor, MI.

Rigo, Armin. 2004. Representation-based just-in-time specialization and the Psyco prototype for Python. In Nevin Heintze and Peter Sestoft, editors, *Proceedings of the 2004 ACM SIGPLAN Workshop on Partial Evaluation and Semantics-based Program Manipulation*, pages 15–26, Verona, Italy.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.

Simard, Michel, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais, and Arne Mauser. 2005. Translating with non-contiguous phrases. In *Proceedings of HLT/EMNLP 2005*, pages 755–762, Ann Arbor, MI.

Stolcke, Andreas. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 901–904, Denver, CO.

Thayer, Ignacio, Emil Ettelaie, Kevin Knight, Daniel Marcu, Dragos Stefan Munteanu, Franz Joseph Och, and Quamrul Tipu. 2004. The ISI/USC MT system. In *Proceedings of IWSLT 2004*, pages 59–60, Kyoto, Japan.

Tillmann, Christoph. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004*, pages 101–104. Companion volume, Boston, MA.

Tillmann, Christoph and Tong Zhang. 2005. A localized preduction model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 557–564, Ann Arbor, MI.

Wu, Dekai. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 152–158, Santa Cruz, CA.

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

Xia, Fei and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of the Twentienth International Conference on Computational Linguistics (COLING)*, pages 508–514, Geneva, Switzerland.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the ACL*, pages 523–530, Toulouse, France.

Zens, Richard and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of HLT-NAACL 2004*, pages 257–264, Boston, MA.