Preference Grammars and Decoding Algorithms for Probabilistic Synchronous Context Free Grammar Based Translation.

Ashish Venugopal

CMU-LTI-09-010

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Stephan Vogel, Chair
Noah A. Smith
Alex Waibel
Kevin Knight, University of Southern California

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Abstract of the Dissertation

Preference Grammars and Decoding Algorithms for Probabilistic Synchronous Context Free Grammar Based Translation.

by

Ashish Venugopal

Doctor of Philosophy in Language Technologies

Language Technologies Institute

Carnegie Mellon University 2008

Professor Stephan Vogel, Chair

Probabilistic Synchronous Context-free Grammars (PSCFGs) [Aho and Ullmann, 1969, Wu, 1996] define weighted transduction rules to represent translation and reordering operations. When translation models use features that are defined *locally*, on each rule, there are efficient dynamic programming algorithms to perform translation with these grammars [Kasami, 1965]. In general, the integration of non-local features into the translation model can make translation NP-hard, requiring decoding approximations that limit the impact of these features.

In this thesis, we consider the impact and interaction between two non-local features, the n-gram language model (LM) and labels on rule nonterminal symbols in the Syntax-Augmented MT (SAMT) grammar [Zollmann and Venugopal, 2006]. While these features do not result in NP-hard search, they would lead to serious increases in wall-clock runtime if naïve dynamic programming methods are applied.

We develop novel two-pass algorithms that make strong decoding approximations during a first pass search, generating a hypergraph of sentence spanning translation

derivations. In a second pass, we use knowledge about non-local features to explore the hypergraph for alternative, potentially better translations. We use this approach to integrate the n-gram LM decoding feature as well as a non-local syntactic feature described below.

We then perform a systematic comparison of approaches to evaluate the relative impact of PSCFG methods over a strong phrase-based MT baseline with a focus on the impact of n-gram LM and syntactic labels. This comparison addresses important questions about the effectiveness of PSCFG methods for a variety of language and resource conditions. We learn that for language pairs that exhibit long distance reordering, PSCFG methods deliver improvements over comparable phrase-based systems and that SAMT labels result in additional small, but consistent improvements even in conjunction with strong n-gram LMs.

Finally, we propose a novel approach to use nonterminal labels in PSCFG decoding by extending the PSCFG formalism to represent hard label constraints as soft preferences. These preferences are used to compute a new decoding feature that reflects the probability that a derivation is syntactically well formed. This feature mitigates the effect of the commonly applied *maximum a posteriori* (MAP) approximation and can be discriminatively trained in concert with other model features. We report modest improvements in translation quality on a Chinese-to-English translation task.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This dissertation concludes my time at Carnegie Mellon, the institution that has granted me every single one of my educational degrees. In the past eleven years I have learned much more than three commencements can possibly reflect, mostly from my exceptional faculty, colleagues and friends, but also from the city Pittsburgh, which will always be my home away from home. I have many people to thank for their support.

First and foremost, I am grateful to Stephan Vogel, who introduced me to machine translation and showed me how exciting research could be. From the early days of the NIST evaluations, when scores seemed to increase every week, to the challenges of designing and pursuing a research agenda, Stephan provided me the personal and professional support that I needed to succeed. This dissertation has been much improved by the diligent and detailed feedback of my committee—Noah A. Smith, Alex Waibel and Kevin Knight. I thank them especially for their insights into both the ideas and the way in which they are presented here.

I owe special thanks to Andreas Zollmann, with whom I have shared the trials and tribulations of the past several years. His confident yet simultaneously critical approach to important questions in research and every-day life have made these years as enjoyable as they have been intellectually stimulating. I am particularly grateful to my fellow graduate students in the ISL lab, Joy Zhang and Kornel Laskowski, who lent their humor to the most stressful of situations and their intellect to the most humble of ideas.

A lot of what I learned in graduate school, was not learned in school. For these lessons, I thank Benjamin Billings, Aris Winger and Michael Maxim, I treasure my friendship with them.

I owe my pursuit of graduate school to my undergraduate mentors, Mark Stehlik and Peter Lee, who created opportunities for me at every step in this long journey. While their efforts to support me have been extraordinary, I am certain that all of their students feel the same way.

My family has been extraordinarily supportive of my efforts. I have been exceptionally fortunate to have my grandparents appreciate and attend graduations and award ceremonies at Carnegie Mellon. To my parents, who have spent days and nights worrying and wondering about machine translation on my behalf, I dedicate this work.

My wife, Priyanka, with whom I have shared five years of excitement, learning, growth and love, gave me confidence in the most trying times of this process. Her emotional and intellectual support is the mortar that binds the chapters of this dissertation together and the muse that kept the words flowing. Thank you.

CHAPTER 1

Introduction

Statistical machine translation defines the task of automatically translating a sentence f in a source language into a sentence e in a target language via techniques that take advantage of commonly available resources and generalize easily to new language pairs. The availability and development of large parallel corpora, inexpensive hardware, efficient machine learning techniques and well defined automatic evaluation measures, have lead to the dominance of *statistical* machine translation as the primary approach to the machine translation task.

Statistical machine translation (SMT) systems often consist of two separate, but mathematically related components. In the *training* component, parallel and monolingual corpora are used to estimate the parameters of a *translation model*, representing structured, weighted operations that transform source language sentences into target language sentences. Well designed translation models capture both the translation of source language words into target language words as well as the relative reordering of target words, ensuring that the meaning of the source sentence is accurately and fluently conveyed in the target language.

In the second component, referred to as *decoding* in the literature, new source language sentences are translated by selecting the most likely translation according to the translation model. Decoding is effectively a search through the space of possible translations licensed by the operations represented in the translation model. In statistical machine translation, like in many natural language tasks, this search is

conducted in a space that is exponentially large relative to the length of the input.

When strong independence assumptions that facilitate local decision making and sub-structure sharing are introduced into the statistical translation model, efficient *dynamic programming* solutions are available for decoding. While these independence assumptions lead to efficient search, they are often inadequate to explain the generation of natural language. Relaxing these independence assumptions to take non-local information into account during search holds the potential for improvements in translation quality but can lead to a dramatic explosion in the search space.

In this dissertation we consider the use of non-local information to improve translation quality. We focus our efforts within a particular class of translation model that has the potential to efficiently and effectively represent the long-distance reordering effects that often occur when translating natural language. We are particularly interested in non-local information that can be used to improve the fluency of the target language output. We propose novel decoding algorithms that approximately integrate non-local information in ways that improve runtime and translation quality, and we empirically evaluate the impact of non-local features in a large scale systematic comparison of approaches.

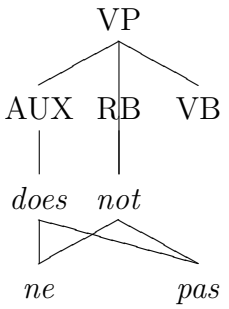
1.1 Background

We focus on a particular class of translation models that represent translation operations in the form of Probabilistic Synchronous Context Free Grammar (PSCFG) *rules*. We will refer to approaches that use the PSCFG formalism as *grammar* based approaches.

Figure 1.1 contains example rules from a sampling of translation models that take advantage of the PSCFG formalism. In each case, a single rule represents the

$\mathrm{X} \rightarrow ne\ \mathrm{X}_1\ pas\ \#\ does\ not\ \mathrm{X}_1 : w$ Chiang [2005]

$\mathrm{VP} \rightarrow ne\ \mathrm{VB}_1\ pas\ \#\ does\ not\ \mathrm{VB}_1 : w$ Zollmann and Venugopal [2006]



Galley et al. [2004]

Figure 1.1: Example rules from a sampling of PSCFG approaches [Chiang, 2005, Zollmann and Venugopal, 2006, Galley et al., 2004]. In each approach, the French words *ne* and *pas* are translated in the context of source words in between *ne* and *pas*. These rules are automatically learned from word aligned corpora and assigned a stochastic weight w.

translation of the source language French words *ne* and *pas* into the target language English words *does not*, in the context of source words in between *ne* and *pas*.

Unlike their purely lexical phrase-based predecessors like Koehn et al. [2003], Och and Ney [2004], PSCFG rules perform translation and reordering within a single operation, allowing these decisions to be made with more contextual information. For example, the translation of the word *ne* in our example above is made in the context of *pas* which could be separated from *ne* by many source words. Each rule is associated with a weight w, which reflects the translation model's confidence in the rule being applied to translate its source words. Source sentences are translated by applying rules recursively to transform source input into a *derivation* with an associated target translation.

Figure 1.2 shows how the example rules that follow Zollmann and Venugopal [2006] in Figure 1.1 compose to form derivations. In this example, constraints en-

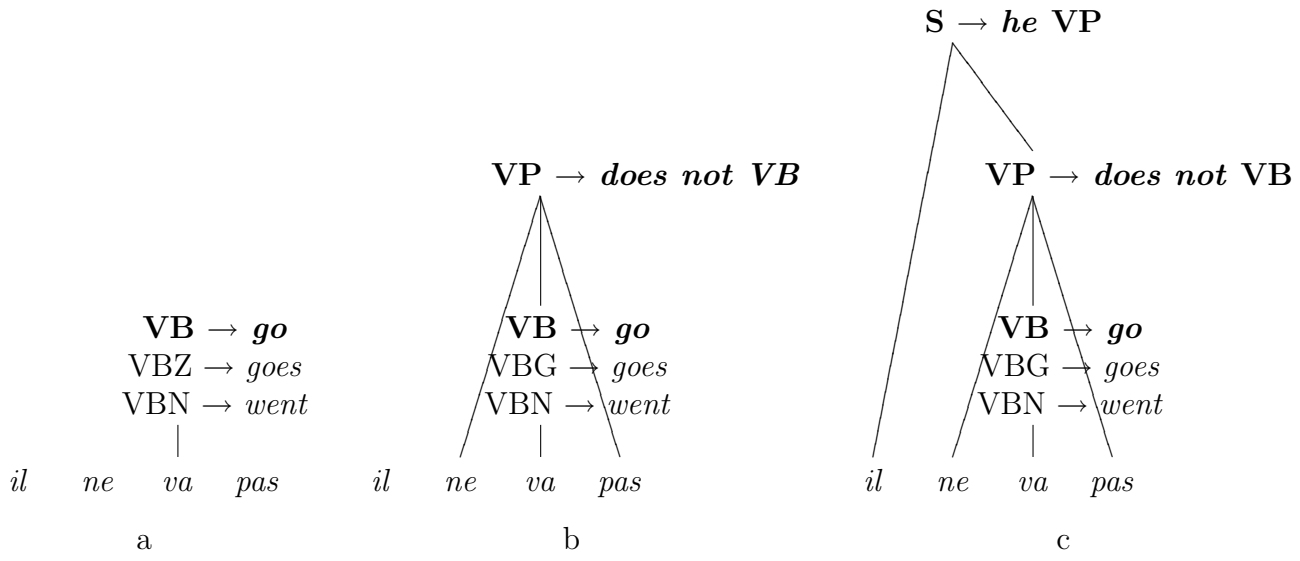


Figure 1.2: The translation of a French source sentence *il ne va pas* using rules from Zollmann and Venugopal [2006], where rules are applied during decoding in Figures a, b, c to form a derivation. In this example, multiple alternatives exist for the translation of the word *va*, each alternative producing a different target output. Rules that successfully compose to form a sentence spanning derivation are in bold.

forced by nonterminal labels cause the decoder to find a fluent translation, discarding alternative derivations that produce translations such as *he does not went*.

When the translation model relies only on *local* information, like weights w associated with each rule, there are efficient polynomial time dynamic programming solutions to select the most likely derivation that translates a source sentence.

In practice however, machine translation systems introduce additional features into the translation model that use non-local information to complement the decoder's search towards fluent target output. The locality of features in a translation model is a function of dynamic programming algorithm used for translation. Dynamic programming algorithms are used to solve problems that decompose into sub-problems whose solutions can be efficiently found and re-used towards solving the larger prob-

lem. Non-local features introduce dependencies between these sub-problems. For some non-local features, like those that we consider here, the definition of the sub-problem can be extended to accommodate these non-local features, ensuring that sub-problems can still be solved correctly. Extending the definition of the sub-problem has the consequence of creating *more* sub-problems that need to be solved [1].

In this work, we will be using the bottom-up CYK decoding algorithm [Kasami, 1965] to translate source sentences with PSCFG rules. This algorithm aims to find the lowest weighted derivation that translates a source sentence using a PSCFG grammar. The sub-problems in this algorithm correspond to finding the lowest weighted derivations that translate sub-spans of the input sentence.

We consider features to be local for this algorithm when they are defined on each rule and do not introduce dependencies or constraints across rules in a derivation. Under this definition of locality, we will consider two approaches to using non-local information. First, we consider the integration of *n-gram* language models (LMs) and second, the use of nonterminal *labels* on PSCFG rules during search.

N-gram language models define a distribution $p_{LM}(e)$. For PSCFG decoding, this makes the selection of a rule in a derivation dependent on derivations that translate neighboring sub-spans. In our example from Figure 1.2, an n-gram LM would introduce additional weights that measure the likelihood of generating target sequences *does not go* versus *does not went*. The decoder must consider these weights when making decisions about which rules to select.

The use of multiple nonterminal labels also serves to propagate non-local information during decoding. In our example, nonterminal labels like VP and VB, which are Penn Treebank categories [Marcus et al., 1993] that represent syntactic behavior, constrain the the rules that can be used for form a derivation. The rule VP $\rightarrow$ $\langle$

[1] This approach is only feasible when the resulting number of sub-problems does not grow exponentially as a function of sentence length.

ne VB *pas* # *does not* VB ⟩, requires a label of VB as input, and thus restricts the search to produce *does not go*. These labels summarize the syntactic behavior of the translation that they represent, allowing this information to constrain the rules that can be used to extend the derivation.

Both of these approaches have a significant impact on search space that the decoder must explore. The CYK decoding algorithm [Kasami, 1965] to perform decoding with PSCFG rules and an n-gram LM has a runtime of:

$$\mathcal{O}(|f|^3|[|\mathcal{N}||\mathcal{T}|^{2(n-1)}]^K) \tag{1.1}$$

where $|f|$ is the length in words of the source sentence to be translated, $\mathcal{N}$ is the set of nonterminal labels in the grammar, $\mathcal{T}$ is the set of target terminals that can be output by the grammar, n is the order of the n-gram language model, and K is the maximum number of nonterminal symbols on the right-hand-side of each rule in the grammar. [Chiang, 2007]. In this dissertation we work with grammars that are restricted to at most two right-hand-side nonterminal symbols[2] therefore $K = 2$.

The constant factor (with respect to f) in Equation 1.1 arises from the increase in the number of sub-problems that we must now solve when using these non-local features. Sub-problems are represented by *items* in the dynamic programming algorithm. Using notation from Chiang [2007], the corresponding item structure to perform search with a n-gram LM and nonterminal labels is:

$$[X, i, j, \tilde{e}] : w \tag{1.2}$$

where X is the nonterminal label of a derivation, i, j define a span in the source sentence, and $\tilde{e}$ maintains context required to apply the n-gram language model to

[2]There are efficient methods to transform most rules with more than two nonterminal symbols into equivalent binarized rules with at most two right-hand-side nonterminal symbols [Zhang et al., 2006]

score derivations. Under the *maximum a posteriori* approximation (MAP) that is often used in SMT [3] we can discard derivations of lower weight that share this item structure. Integrating non-local features into the model corresponds to adding more elements (like $\tilde{e}$) to the item structure. In practice, this approach requires pruning to limit the number of items produced, thereby limiting the potential impact of non-local features.

Instead we might choose to perform search under a simpler model with only local features and generate a list of alternative derivations that we can efficiently score under the more complex model. If we chose to ignore the n-gram LM and use a grammar with only a single nonterminal symbol we could use the item structure $[i, j]$ with a corresponding runtime of $\mathcal{O}(|f|^3)$. In practice, the alternative derivations generated by this simpler model represent a very small fraction of the complete search space, yielding limited potential for improvements during rescoring [Chiang, 2007].

In this dissertation, we propose novel approaches to better integrate *n-gram* language models and nonterminal *labels* into the PSCFG translation model and decoding search strategy. We evaluate our work primarily with the Syntax-Augmented MT (SAMT) grammar from Zollmann and Venugopal [2006]. Our techniques take advantage of knowledge regarding the structure of these non-local features, suggesting efficient search and modeling approximations to improve translation runtime and quality. We also conduct a systematic comparison of approaches, where we evaluate the relative impact of PSCFG methods when using strong n-gram LMs and syntactic labels on a variety of language and resource conditions.

[3] Selecting the most likely translation, marginalizing over alternative derivations that produce the same translation instantiates an NP-hard inference task for even simple word-based models [Knight, 1999]

1.2 Thesis Outline and Contributions

- In Chapter 2, we describe the motivation behind the PSCFG approach to machine translation. We discuss two approaches to grammar based decoding, the *hierarchical* approach from Chiang [2005] (Hiero) and the *syntax-augmented* approach from Zollmann and Venugopal [2006] (SAMT) that differ only in their use of nonterminal labels. We will use systems built from these grammars in the following chapters.

- Chapter 3 develops two multi-pass decoding algorithms to efficiently introduce n-gram language models into a PSCFG decoder. These algorithms make strong approximations in a first pass search and generate a pruned hypergraph representation of the search space that can be explored with non-local features to select better translations. We evaluate our algorithms on a small, but challenging Chinese-to-English translation task using the SAMT grammar.

- In Chapter 4, we answer important questions about the impact of PSCFG methods, specifically studying the impact of nonterminal labels on translation quality. We conduct a large-scale systematic study across multiple language pairs that ultimately addresses the question as to whether grammar based approaches (SAMT and Hiero), can deliver improvements over phrase-based approaches under comparable conditions.

- In Chapter 5 we propose a novel grammar formalism and associated translation model feature to take advantage of nonterminal labels without aggravating the impact of the common MAP approximation that is often used in machine translation. Our approach transforms hard syntactic constraints into soft preferences that are used to estimate a syntactic consistency feature, whose weight can be tuned to improve translation quality. We evaluate our work on small

and medium sized Chinese-to-English translation task.

CHAPTER 2

Background: Statistical Machine Translation and PSCFG Approaches

In this chapter we provide the background to understand our contributions to the field of PSCFG based translation. Starting with an introduction to statistical machine translation, we discuss the motivations for grammar based approaches and survey the major research directions in this field. We pay particular attention to Hierarchical Phrase-based translation (Hiero) [Chiang, 2005] and its labeled extension Syntax Augmented Machine Translation (SAMT) [Zollmann and Venugopal, 2006]. Systems built from these grammars will be used in the following chapters.

2.1 Statistical Machine Translation

SMT suggests an approach to natural language translation based on machine learning approaches, where statistical models are estimated from natural language corpora and applied to translate new source sentences. Comprehensive surveys and tutorials of the field are available in Lopez [2008], Koehn [2007]. Here we survey the literature that motivates our work with PSCFG methods, starting with a brief overview of the major components in SMT systems.

Early work in SMT [Brown et al., 1990, 1993] followed a noisy channel approach, modeling the conditional probability of generating a sequence of target language words

e from a sequence of source language words f as:

$$p(e|f) = \frac{p(f|e)p(e)}{p(f)} \tag{2.1}$$

The model defined above decomposes neatly into a *translation model*: $p(f|e)$ and a *language model*: $p(e)$, whose parameters can be estimated from bilingual parallel data and monolingual target language data, respectively. Brown et al. [1993] developed techniques to estimate $p(f|e)$ from sentence-aligned parallel corpora via an Expectation Maximization procedure [Dempster et al., 1977], generating, among other intermediate models, the most likely word-to-word alignment across source and target sentences. The language model component $p(e)$, which assigns a probability to sequences of target language words can be estimated from monolingual data.

Translating a new sentence f can be defined as a search task:

$$\hat{e} = \arg\max_{e} p(e|f) \tag{2.2}$$

$$= \arg\max_{e} p(f|e)p(e) \tag{2.3}$$

Equation 2.2 selects the target sequence $\hat{e}$ that maximizes $p(e|f)$. It is unnecessary to calculate the normalization constant $p(f)$ during search since its value is constant for all e that translate f. In practice, most SMT systems introduce additional independence assumptions into $p(e|f)$ to facilitate efficient search via *dynamic programming*. These independence assumptions correspond to restrictions in the operations that can be taken to translate a source sentence. A sequence of these translation operations is referred to as the *derivation* that creates a target translation. While there are efficient techniques to select the most likely derivation for the models discussed in this work, selecting the most likely *translation* is NP-hard for word-based [Knight, 1999] and grammar based translation models [Casacuberta and de la Higuera, 2000,

Sima'an, 2002]. Most systems therefore use the maximum a posteriori (MAP) approximation, selecting the most likely derivation under the model and outputting the associated translation:

$$\hat{e} = \arg\max_{e} \sum_{d} p(e, d \mid f) \tag{2.4}$$

$$\approx \arg\max_{e} \max_{d} p(e, d \mid f) \tag{2.5}$$

Och and Ney [2002] generalize the model from Brown et al. [1990] using the maximum entropy framework [Berger et al., 1996b] to directly model the posterior probability $p(e|f)$ as:

$$p(e, d|f) = \frac{\prod_{i=1}^{m} h_i(e, d|f)^{\lambda_i}}{Z(\lambda)} \tag{2.6}$$

where $h_i(d, e|f)$ are feature functions that model translation quality and λ_i are corresponding weights that assign relative importance to these features. Under this framework, $p(f|e), p(e)$ can be introduced as features in h. An important advantage of this framework is that model parameters λ can be discriminatively trained on development data to maximize either model likelihood or an external evaluation measure.

Many approaches have been proposed to evaluate the quality of machine translation output, traditionally requiring human evaluators who are knowledgeable in the target language to compare SMT output to a human produced *reference translation* on two dimensions. The first is adequacy, which measures the degree to which information in the reference is conveyed in SMT output. The second is fluency, which measures how fluently the MT output reads [White et al., 1994]. While human evaluations are the ultimate standard to judge SMT output, they are expensive and time-consuming to conduct on scales large enough to inform research in the field.

As an alternative to human evaluation, several automatic evaluation measures have been proposed, each considering alternative methods to compare SMT with reference output [Vidal, 1997, Papineni et al., 2002, Doddington, 2002]. The most widely accepted measure is the BLEU (BiLingual Evaluation Understudy) score [Papineni et al., 2002], which considers word sequence matches between SMT output and potentially multiple reference translations. BLEU calculates n-gram precisions—the number of n-grams that occur in both the MT output and the corresponding reference translations as a fraction of the total number of n-grams hypothesizes in the MT output. The n-gram precision is calculated as:

$$p_n = \frac{\sum_{C \in Candidates} \sum_{g_n \in C} Count_{clip}(g_n)}{\sum_{C \in Candidates} \sum_{g_n \in C} Count(g_n)}$$

where C is a MT output sentence from the set of outputs ($Candidates$) for a test corpus, g_n is a sequence of n target words, $Count(g_n)$ is the number of times g_n occurs in C, and $Count_{clip}(g_n)$ is the maximum number of times g_n occurs in both C and a set of reference translations. The precision component of the score is balanced by a recall component that penalizes translation output with fewer words than the reference translation. BLEU is calculated as:

$$BLEU = BP \cdot e^{1/N \sum_{n=1}^{N} \log p_n} \tag{2.7}$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1-r/c} & \text{if } c \leq r \end{cases} \tag{2.8}$$

where c is the combined length of all the SMT outputs in $Candidates$ and r is the length of the corresponding reference translations. When multiple references are available, the reference of closest length is selected for each c. This score is efficient to calculate on large corpora and Och [2003] describes a procedure called Minimum Error Rate Training (MERT) to learn λ such that the BLEU score is optimized on

development corpus. In this work, the BLEU metric will be our primary measure of translation quality, with $N = 4$.

While the BLEU metric has been successful in driving the whole field of machine translation forward, the emergence of syntax based approaches has renewed interest in metrics that can reward correct sentence structure, like Banerjee and Lavie [2005] rather than surface level n-gram matches. A survey of the most common automatic evaluation metrics and their respective correlation to human judgement on a variety of languages can be found in Callison-Burch et al. [2007]. With a brief overview of SMT completed, we now discuss techniques to perform the search in Equation 2.4.

2.2 Motivating PSCFG Approaches

While the model defined in Equation 2.6 explains how to score a candidate translation, it does not define how to generate and search through the space of possible translation alternatives. The space of translation alternatives is exponential in the number of words in the input sentence. Enumerating all possible alternative translations and scoring each one according to the model is not computationally tractable. This is a common challenge in many natural language processing tasks and is often solved by introducing additional structure to the search task, typically corresponding to strong model independence assumptions. For example, efficient dynamic programming algorithms exist to find the most likely derivation of hidden states for a Hidden Markov Model (HMM) when the model includes only *local* transition and emission features [Rabiner, 1989]. The most probable derivation from applying a grammar to an input sentence can be efficiently found when the probability decomposes at each rule used in the derivation [Kasami, 1965, Aho and Ullmann, 1969]. Most popular SMT approaches follow a similar strategy, the optimal derivation can be efficiently found when $p(e, d|f)$ decomposes into local (partial sentence) translation and reorder-

ing decisions. The design of these local translation and reordering decisions and the independence assumptions that they correspond to, have a large impact on translation quality and runtime. The PSCFG approach to SMT generates the full sentence translation via the composition of weighted grammar rules [Aho and Ullmann, 1969], where most features in h can be computed locally, on a per-rule basis. We now motivate the PSCFG approach by briefly surveying the development of word and phrase based translation models.

Brown et al. [1993] define a translation model that makes strong independence assumptions about how translation occurs. These independence assumptions allow the model's parameters to be efficiently estimated from parallel corpora and suggests the basic operations that are applied to translate new sentences. Knight and Al-Onaizan [1998] provide a finite state interpretation of the models proposed in Brown et al. [1993] (upto Model 3) to perform decoding using the well understood weighted automata framework. That work, along with Wang and Waibel [1997], Tillmann et al. [1997], propose decoding algorithms to translate source sentences based on the translation model parameters estimated in Brown et al. [1993].

Even with these relatively simple models, there are computational challenges that result from modeling the reordering effects that occur across languages. Under Brown et al. [1993], words are translated and permuted according to a position based distortion model. Generating the full space of possible permutations for an input sentence is computationally challenging [Och, 2003] and selecting the most likely translation considering all possible permutations is NP-hard [Knight, 1999]. In practice, most systems constrain these word order permutations to permit only local, short distance reordering [Berger et al., 1996a].

These word-based models have significant limitations. They are unable to capture non-literal phrasal translation, would require many-to-many word alignments. Word-

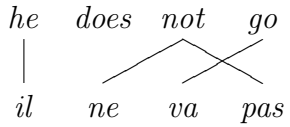


Figure 2.1: Word alignment for an example French-English sentence pair. Alignments between words are represented by links across languages.

based models are also too poorly parametrized to produce the complex reordering patterns that exist across natural languages. Phrase-based models were introduced [Och et al., 1999] and developed [Koehn et al., 2003, Marcu and Wong, 2002, Och and Ney, 2004, Venugopal et al., 2003] to directly address these issues. For these approaches, the fundamental unit of translation is a word sequence or *phrase*. Using phrase-based models, the input sentence is segmented into phrases where each phrase is translated and permuted to produce the final translation. Simple and relatively short phrases can capture non-literal translation, local word reordering and insertion or deletion of source and target words [Koehn et al., 2003]. Most phrase-based approaches [Och and Ney, 2002, Koehn et al., 2003] identify bilingual phrase pairs based on the word alignments from Brown et al. [1993], while others, like Marcu and Wong [2002], propose generative models to learn phrase translations directly from parallel corpora.

Since the phrase-based approach provides the foundation for the PSCFG methods used in this thesis, we follow a simple example to highlight the advantages and disadvantages of phrase-based translation. Consider the word-aligned French-English sentence pair in Figure 2.1.

The method in Och et al. [1999], Koehn et al. [2003] extracts all bilingual phrase pairs that do not have words aligned outside the phrase pair. The phrase pairs in Figure 2.2 would be extracted for the sentence pair in Figure 2.1.

For each of these phrase pairs, identified by their source words f and target words

$$
\begin{aligned}
\textit{il} &\ \# \ \textit{he} \\
\textit{il} &\ \# \ \textit{he does} \\
\textit{va} &\ \# \ \textit{go} \\
\textit{ne va pas} &\ \# \ \textit{not go} \\
\textit{il ne va pas} &\ \# \ \textit{he does not go}
\end{aligned}
$$

Figure 2.2: Phrase pairs as extracted by Koehn et al. [2003] for the example word-aligned sentence from Figure 2.1.

e the following features h_i are computed [Koehn et al., 2003, 2004]. We use named feature functions instead of indices for clarity:

- $h_{trans} = p(f|e) = \frac{cnt(f,e)}{\sum_f cnt(f,e)}$: a translation model probability estimated by relative frequency from rule occurrence counts (cnt) in the training corpora.
- $h_{lex} = p(f|e,a) = \prod_{i=1}^{n} \frac{1}{\{|j|(i,j) \in a\}} \sum_{\forall (i,j) \in a} w(f_i|e_j)$: a lexical weight computed by considering word-to-word translation probabilities $w(f|e)$ where a is a word alignment between word positions $i = 1, \cdots, n \in f$ and $j = 0, 1, \cdots, m \in e$.
- $h_{len} = |e|$: a factor that allows the decoder to control the length of the output translation.
- $h_{count} = 1$: a factor that allows the decoder to control the number of phrases used in the output translation.

Each of the features described above decomposes locally. They can be precomputed for each phrase pair used in a derivation. Koehn et al. [2004] also use additional features that are conditioned in reverse, i.e $p(e|f), p(e|f,a)$, while the following fea-

tures which are computed based on the derivation of phrases used to translate a sentence:

- $h_{LM} = p(e)$: the language model feature as proposed in Brown et al. [1990].
- $h_{dist} = d(a_i - b_{i-1}) = 1^{|a_i - b_{i-1} - 1|}$: a simple distortion model that penalizes reordering, where a_i denotes the start position of the foreign phrase that was translated into the i'th English phrase and b_{i-1} denotes the end position of the foreign phrase translated into the $(i-1)$'th English phrase.

Phrase-based translation has been, until very recently, the dominant approach to machine translation. Open-source decoders such as Pharaoh [Koehn et al., 2004] and its successor Moses [Koehn, 2007], have been used to build machine translation for a variety of languages [Koehn and Monz, 2005]. While local reordering operations are well represented within each phrase pair, empirical results show that longer and more complex reordering effects are required for effective translation [Fox, 2002, Hwa et al., 2002, Wellington et al., 2006]. To account for these effects, phrase-based models allow phrases to be translated, or distorted out of order and introduce additional features to evaluate distortion operations.

There has been significant work in the field to model distortion for phrase-based approaches. Koehn et al. [2003], Vogel et al. [2003] introduce additional features that penalize long distance reordering, independent of the phrases involved in the reordering, while Och and Ney [2004], Zens and Ney [2006a] take into account the lexical content of each phrase. Nießen and Ney [2004], Xia and McCord [2004], Wang et al. [2007] avoid generating permutations during decoding, they pre-process the input sentence to reflect target language word order.

The approaches above rely on the decoder, or a pre-processing step, to propose alternative permutations of the source sentence. The PSCFG approach to SMT allows

these operations to be learned and parameterized from training data. In Section 2.3 we review the PSCFG formalism and describe specific instantiations that have been proposed in the literature, followed by a discussion of the two specific approaches, Chiang [2005] and Zollmann and Venugopal [2006], used in this work. In Section 2.6 we discuss decoding with PSCFGs.

2.3 Grammar Based Approaches

The PSCFG formalism can be viewed as an extension of the phrase-based approach, using nonterminal symbols, as in monolingual parsing, to extend the context that is available to make translation and reordering decisions. PSCFGs are defined in Chiang [2005] by a source terminal set (source vocabulary) $\mathcal{T}_S$, a target terminal set (target vocabulary) $\mathcal{T}_T$, a shared nonterminal set $\mathcal{N}$ and *rules* of the form:

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

- $X \in \mathcal{N}$ is a left-hand-side (lhs) nonterminal symbol
- $\gamma \in (\mathcal{N} \cup \mathcal{T}_S)*$ is a sequence of nonterminals and source terminals.
- $\alpha \in (\mathcal{N} \cup \mathcal{T}_T)*$ is a sequence of nonterminals and target terminals.
- $\sim$ enforces a one-to-one mapping between co-indexed nonterminals in γ and α.
- $w \in [0, \infty)$ is a non-negative real-valued weight assigned to the rule. In practice, we define w as part of a log-linear model that includes several rule features with learned weights i.e $w = \prod_{i=1}^{m'} h_i(r)^{\lambda_i}$ where $m' \leq m$, which is the total number of features in the model.

For visual clarity, example PSCFG rules in this thesis will use the # character to separate the source side of the rule γ from the target side α. PSCFG rules trans-

form input source words into the target language words via intermediate nonterminal symbols, similar to monolingual parsing. Decoding with PSCFG rules is discussed in Section 2.6. Aho and Ullmann [1969], Melamed [2003], Wu [1996] present complexity analysis for various PSCFG approaches and show efficient dynamic programming solutions for parsing with these grammars. PSCFG approaches vary in the restrictions placed upon their formal components. The following approaches all use a single nonterminal label, i.e $\mathcal{N} = \{X\}$:

- Wu [1997] separate purely nonterminal based reordering rules from lexicalized translation rules.
- Chiang [2005] allows up to two nonterminals in lexicalized rules that include both terminal and nonterminal symbols.
- Watanabe et al. [2006] restricts the rules from Chiang [2005] to those that have Chomsky-Normal Form.

The hierarchical nature of this formalism is similar to our understanding of syntactic structure and several approaches have been suggested to take advantage of this. The approaches below use syntactically derived nonterminal labels:

- Zhang and Gildea [2005] extend Wu [1997] by lexicalizing the nonterminal labels.
- Zollmann and Venugopal [2006] extend Chiang [2005] to use labeled nonterminal symbols. Labels are identified based on target language syntactic phrase structure parse trees.
- Galley et al. [2004, 2006] uses target language syntactic structure to inform the identification of rules. These approaches are built upon the more powerful *tree transducer* grammar formalism, but in practice the rules generated by these approaches can be represented by PSCFG rules.

Each of the approaches above correspond to alternative parameterizations of the PSCFG formalism. We use two PSCFG grammars in our work; Hiero [Chiang, 2005], which uses a single nonterminal label and SAMT [Zollmann and Venugopal, 2006], which uses a large syntactically motivated nonterminal set. These approaches differ primarily in their use of nonterminal labels. We describe the grammar extraction procedures of these two approaches in Section 2.4, 2.5 and provide background to understand our decoding contributions in Section 2.6.

2.4 Hierarchical PSCFG

Chiang [2005] describes a procedure for learning a *hierarchical* phrase-based model (Hiero) from word-aligned parallel corpora, generating a PSCFG with a single nonterminal label. Initial phrase pairs are identified following Koehn et al. [2003] and initial phrases are limited to have up to β_{len} source words. Initial phrases are assigned a generic nonterminal label X, forming initial *rules*. These rules are used as a lexical basis to form rules with *right-hand-side* nonterminal symbols using the procedure below: For each *rule*:

$$\mathrm{X} \rightarrow f_1 \dots f_m \ \# \ e_1 \dots e_n$$

for which is an *initial rule*:

$$\mathrm{X} \rightarrow f_i \dots f_u \ \# \ e_j \dots e_v$$

where $1 \leq i < u \leq m$ and $1 \leq j < v \leq n$, a new rule can be generated that has the form:

$$\mathrm{X} \rightarrow f_1 \dots f_{i-1} \ X_k \ f_{u+1} \cdots f_m \ \# \ e_1 \dots e_{j-1} \ X_k \ e_{v+1} \dots e_n$$

where nonterminal indices in γ, α are co-indexed by k. This generalization procedure can be performed recursively to create rules with multiple nonterminal symbols. Chi-

ang [2005] imposes the following grammar restrictions; rules are limited to at most two, non-adjacent nonterminals, rules must have at least one aligned word and when multiple initial phrases contain the same set of word-to-word alignments, only the smallest phrase is retained. The following initial rules would be extracted for the example French-English parallel sentence in Figure 2.1 (note the removal of the phrase pairs that have the unaligned target word *does*):

$$\begin{aligned}
\mathrm{X} &\rightarrow \textit{il} \ \# \ \textit{he} \\
\mathrm{X} &\rightarrow \textit{va} \ \# \ \textit{go} \\
\mathrm{X} &\rightarrow \textit{ne va pas} \ \# \ \textit{not go} \\
\mathrm{X} &\rightarrow \textit{il ne va pas} \ \# \ \textit{he does not go}
\end{aligned}$$

resulting in the following rules with right-hand-side nonterminal symbols:

$$\begin{aligned}
\mathrm{X} &\rightarrow \mathrm{X}_1 \ \textit{ne va pas} \ \# \ \mathrm{X}_1 \ \textit{does not go} \\
\mathrm{X} &\rightarrow \textit{il ne} \ \mathrm{X}_1 \ \textit{pas} \ \# \ \textit{he does not} \ \mathrm{X}_1 \\
\mathrm{X} &\rightarrow \textit{il} \ \mathrm{X}_1 \ \# \ \textit{he} \ \mathrm{X}_1 \\
\mathrm{X} &\rightarrow \textit{ne} \ \mathrm{X}_1 \ \textit{pas} \ \# \ \textit{not} \ \mathrm{X}_1
\end{aligned}$$

In addition to the automatically identified rules above, two additional rules are added to the grammar to allow the decoder to serially combine translations, similar to phrase-based decoding:

$$\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{S}_1 \ \mathrm{X}_2 \ \# \ \mathrm{S}_1 \ \mathrm{X}_2 \\
\mathrm{S} &\rightarrow \mathrm{X}_1 \ \# \ \mathrm{X}_1
\end{aligned}$$

The first of these two rules is referred to as the "glue" rule. Rule features h for Hiero rules include those from Koehn et al. [2003] (described in Section 2.2), as well as an additional binary feature h_{glue} that allows the decoder to discriminate (using λ_{glue}) between derivations that use the glue rule and those that do not.

2.5 Syntax Augmented PSCFG

Syntax Augmented Machine Translation (SAMT) [Zollmann and Venugopal, 2006] extends Hiero to use nonterminal labels identified using target language parse trees. The motivation for using target language parse trees for labels is similar to Galley et al. [2004]; using target language syntactic labels can constrain translation output towards syntactically well formed target sentences.

Inputs to SAMT rule extraction procedure are tuples: $\langle f, e, Phrases(a, f, e), \pi \rangle$, where f is a source sentence, e is a target sentence, a is a word-to-word alignment associating words in f with words in e, $Phrases(a, e, f)$, are the set of phrase pairs (source and target phrases) consistent with the alignment a [Koehn et al., 2003, Och and Ney, 2004], and π is a phrase structure parse tree of e. π is a phrase structure parse tree that follows the Penn Treebank [Marcus et al., 1993] annotation conventions. The Penn Treebank is a large corpus of human-parsed sentences that is often used to build and evaluate syntactic parsers. In this work, we will generate π using a stochastic parser like Charniak [2000] that has been trained on the Penn Treebank corpus. The constituent labels in π include Part-of-Speech labels like DT for determiners, phrase labels like NP for Noun Phrase, Clause labels like S for a simple declarative clause. See Marcus et al. [1993] for the complete list of labels that can occur in π.

SAMT rule extraction associates each initial phrase pair from $Phrases(a, e, f)$ with a left-hand-side label to form *initial rules*. Labels are assigned based on the

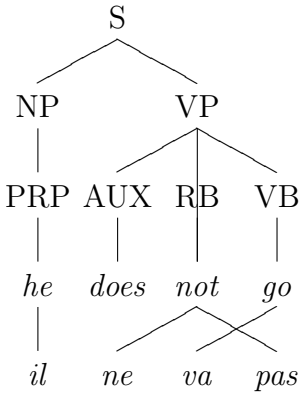


Figure 2.3: Alignment graph [Galley et al., 2004], for a French-English sentence pair.

constituent spanning the target side of the phrase in π. The rule extraction procedure from Chiang [2005] is then applied to generate rules with *labeled* right-hand-side nonterminal symbols.

Consider the example alignment graph (word alignment and target language parse tree Galley et al. [2004]) for the example French-English sentence pair in Figure 2.3. When the target side of the initial phrase pair is spanned by a single constituent in π, the constituent label is assigned as the lhs for the phrase pair. If the target side of the phrase is not spanned by a single constituent in π, SAMT uses the labels of subsuming, subsumed, and neighboring constituents in π to assign an extended label of the form C_1+C_2, C_1/C_2, or C_2\C_1 (similar in motivation to the labels in Steedman [1999]). These labels indicate that the phrase pair's target side spans two adjacent syntactic categories (e.g., *she went*: NP+VB), a partial syntactic category C_1 missing a C_2 at the right (e.g., *the great*: NP/NN), or a partial C_1 missing a C_2 at the left (e.g., *great wall*: DT\NP), respectively. The label assignment is attempted in the order just described. If no label is assignable by either of these three methods, then triple-concatenation is used to create a label of the form C_1+C_2+C_3. If this approach still does not yield a label, the generic label X is assigned. An ambiguity arises for

unary rules $N_1 \rightarrow \cdots \rightarrow N_m$ in π are encountered, such as the NP→PRN subtree in Figure 2.3. In this case, a composite label $N_1 : \cdots : N_m$ is assigned.

Coming back to our example, the initial phrases from Figure 2.2 would be labeled as:

$$\begin{aligned}
\text{PRP:NP} &\rightarrow \textit{il \# he} \\
\text{PRP+AUX} &\rightarrow \textit{il \# he does} \\
\text{VB} &\rightarrow \textit{va \# go} \\
\text{RB+VB} &\rightarrow \textit{ne va pas \# not go} \\
\text{VP} &\rightarrow \textit{ne va pas \# does not go} \\
\text{S} &\rightarrow \textit{il ne va pas \# he does not go}
\end{aligned}$$

The following rules with right-hand-side nonterminal symbols are then produced:

$$\begin{aligned}
S &\rightarrow \text{PRP:NP}_1 \textit{ ne va pas } \# \text{ PRP:NP}_1 \textit{ does not go} \\
S &\rightarrow \text{PRP+AUX}_1 \textit{ ne va pas } \# \text{ PRP+AUX}_1 \textit{ not go} \\
S &\rightarrow \textit{il ne } \text{VB}_1 \textit{ pas } \# \textit{ he does not } \text{VB}_1 \\
S &\rightarrow \textit{il } \text{VP}_1 \# \textit{ he } \text{VP}_1 \\
S &\rightarrow \textit{il } \text{RB+VB}_1 \# \textit{ he does } \text{RB+VB}_1 \\
\text{VP} &\rightarrow \textit{ne } \text{VB}_1 \textit{ pas } \# \textit{ does not } \text{VB}_1 \\
\text{RB+VB} &\rightarrow \textit{ne } \text{VB}_1 \textit{ pas } \# \textit{ not } \text{VB}_1 \\
\text{VP} &\rightarrow \text{RB+VB}_1 \# \textit{ does } \text{RB+VB}_1
\end{aligned}$$

Unlike Chiang [2005], when multiple initial phrases contain the same set of alignment points, all of these alternatives are retained since each one could correspond to a different lhs label. In addition, the SAMT system also retains rules that have no aligned words as long as they have at least one nonterminal symbol, while rules that

have consecutive source nonterminals are discarded. The labeling method described above generates a large number of nonterminal labels, many of them occur very rarely in the training data. As a form of smoothing, the SAMT grammar includes all X labeled rules from Hiero as well. The following additional features, beyond those from Hiero, are introduced by SAMT:

- $h_{lhs}(r) = p(r|\,\text{lhs}(r))$: probability of a rule given its lhs label as in monolingual parsing.
- $h_{lex}(r) = 1$: if the rule has no nonterminals, 0 otherwise.
- $h_X(r) = 1$: if the rule is a Hiero rule.
- $h_{mono}(r) = 1$: if the rule does not reorder its nonterminals, 0 otherwise.
- $h_{rare}(r) = e^{(1/cnt(r))}$: uses the number of times a rule has been seen during training, $cnt(r)$, to allow penalization of derivations that use rare rules.

Since the source and target side of SAMT rules contain nonterminals labels, rule counts are likely to be sparse. To provide further smoothing, each PSCFG rule is stripped of its labels and an additional translation model feature is calculated $h_{transu} = p(\text{ul}(\gamma)|\,\text{ul}(\alpha))$, where the ul function replaces nonterminal labels with.Like Hiero, the SAMT system introduces one glue rule for each unique lhs symbols encountered during in the grammar. Each of these glue rules has $h_{glue} = 1$.

2.6 Decoding with PSCFGs

For clarity, we reformulate the model (Equation 2.6) and search task (Equation 2.4) to refer to the components of the PSCFG formalism and the features that are used in this work. Given a source sentence f and a PSCFG G, the translation task can be

expressed similarly to monolingual parsing with a PCFG. We aim to find the most likely derivation d of the input source sentence and read off the English translation, identified by composing α from each rule used in the derivation. This search for the most likely translation under the MAP approximation can be defined as:

$$\hat{e} = \text{tgt}\left(\underset{d \in \mathcal{D}(G):\text{src}(d)=f}{\arg\max} p(d)\right) \tag{2.9}$$

where $\text{tgt}(d)$ is the target-side yield of a derivation d, and $\mathcal{D}(G)$ is the set of G's derivations. Using an n-gram language model to score derivations and rule labels to constrain the rules that form derivations, we can define $p(e|f)$ as a log-linear model in terms of the rules $r \in \mathcal{R}$ used in d as:

$$p(d) = \frac{\left(\prod_{i=1}^{m'} p_i(d)^{\lambda_i}\right) \times p_{\text{LM}}(\text{tgt}(d))^{\lambda_{m'+1}} \times p_{\text{syn}}(d)^{\lambda_m}}{Z(\lambda)} \tag{2.10}$$

$$p_i(d) = \prod_{r \in \mathcal{R}} h_i(r)^{\text{freq}(r;d)}$$

$$p_{\text{syn}}(d) = \begin{cases} 1 & \text{if } d \text{ respects label constraints} \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

The features in $p(d)$ include the n-gram language model p_{LM}, a collection of m' rule feature functions $h_i : \mathcal{R} \rightarrow \mathbb{R}_{\geq 0}$, and a "syntax" feature that, redundantly, requires every nonterminal label to be expanded by a rule with the same nonterminal label on its lhs. $\text{freq}(r; d)$ denotes the frequency of the rule r in the derivation d. Note that $p_{syn}(d)$ can be effectively ignored as defined in Equation 2.11; it corresponds to the constraints specified in the PSCFG formalism where rule nonterminal labels must agree at their shared nonterminals. The Viterbi algorithm for Context Free Grammars (CFGs) [Kasami, 1965] can be used to select the most likely derivation under this model in polynomial time as a function of source sentence length.

As discussed in Section 1.1 (repeated here for clarity), this algorithm proceeds in a bottom-up fashion, selecting the most likely derivation for short spans of the source

sentence and extending them with rules to translate longer spans. Partial derivations are stored in a chart, which is a dynamic programming data structure where partial solutions are efficiently stored and indexed. In order to make optimal local decisions according to the model, each derivation is identified by information that could impact its selection. The set of annotations that identify a derivation is called a chart item. The chart item structure (notation from Chiang [2007]) required to do search based on the model in Equation 2.10 is:

$$[\mathrm{X}, i, j, \tilde{e}] : w \tag{2.12}$$

where X is the nonterminal label of a derivation, i, j define a span in the source sentence, $\tilde{e}$ is the minimal context information required to compute $p_{\mathrm{LM}}(\alpha)$, where α is the target translation associated with the derivation. When derivation d is the highest weighted derivation identified by this chart item, $w = p(d)$. Under the MAP criterion we can discard derivations of lower weight that share the same item structure. The corresponding runtime of a decoder that uses this chart item structure is [Chiang, 2007]:

$$\mathcal{O}\left(|f|^3 \left[|\mathcal{N}||\mathcal{T}_T|^{2(n-1)}\right]^K\right) \tag{2.13}$$

where $|f|$ is the length in word of the source sentence to be translated, $\mathcal{N}$ is the set of nonterminal labels in the grammar, $\mathcal{T}$ is the set of target terminals that can be output by the grammar, n is the order of the n-gram language model, and K is the maximum number of nonterminal symbols in the grammar. In this dissertation we work with grammars that have at most two nonterminal symbols on the right-hand-side of each rule, therefore $K = 2$. We use the open-source decoder described in Zollmann and Venugopal [2006] for the experiments in this thesis.

For the SMT systems considered here (Hiero, SAMT), the constant term in Equa-

tion 2.13 has a significant impact on wall-clock translation time. To reduce translation time *beam search* is used. Chart items are pruned away based on their relative weight compared to the *best* item shares the same X, i, j.

Beam pruning is performed with two parameters, β_n and β_w. β_n is a limit on the number of chart items that have the same X, i, j and β_w is the maximum allowed weight difference between an item and the best item. Following Chiang [2005], the decoder used in this work enforces a reordering limit, set to the same value as the initial phrase length limit β_{len}. Derivations whose source span is more than β_{reo} words can only be extended by glue rules. Pruning of this nature can produce *search errors*; discarded items might have had the potential to participate in a higher weighted derivation than was found after pruning. This kind of pruning reduces the impact that non-local features can have on translation quality.

Chiang [2007] explicitly represents the steps taken by a PSCFG decoder as inference operations [Shieber et al., 1995] as reproduced in Figure 2.4. Each inference operation (Equations 2.14,2.15,2.16) represents inputs in its numerator and outputs in its denominator. Inputs are existing chart items (antecedents items) and grammar rules, outputs are longer chart items (consequent items). The q function is used to retain the elements of the target translation α that are needed to correctly make MAP decisions based on p_{LM}. This algorithm assumes that the source sentence has begin-of-sentence $\langle s \rangle$ and end-of-sentence $\langle \backslash s \rangle$ markers. The inference rules are explored in bottom-up fashion in ascending order of $j - i$ until the Goal item is produced—representing the translation of the complete source sentence. The computational impact of using non-local information can be seen from inference Equation 2.16. This inference operation must be performed for each antecedent item pair: $\langle [X, i_1, j_1, \tilde{e_1}], [Y, i_2, j_2, \tilde{e_2}] \rangle$ that is compatible with the potentially multiple rules that have the form: $Z \rightarrow \langle f_{i+1}^{i_1} X_1 f_{j_1+1}^{i_2} Y_2 f_{j_2+1}^{j}, \cdot \rangle$.

$$\frac{}{X \to \langle \gamma, \alpha \rangle : w} \quad (X \to \langle \gamma, \alpha, w \rangle) \in G \tag{2.14}$$

$$\frac{X \to \langle f_{i+1}^{j}, \alpha \rangle : w}{[X, i, j, q(\alpha)] : wp(\alpha)} \tag{2.15}$$

$$\frac{Z \to \langle f_{i+1}^{i_1} X_1 f_{j_1+1}^{i_2} Y_2 f_{j_2+1}^{j}, \alpha \rangle : w \ , \ [X, i_1, j_1, \tilde{e}_1] : w_1 \ , \ [Y, i_2, j_2, \tilde{e}_2] : w_2}{[Z, i, j, q(\alpha')] : w w_1 w_2 p(\alpha') \quad (\text{where } \alpha' = \alpha\,[\tilde{e}_1/X_1, \ldots, \tilde{e}_2/X_2])} \tag{2.16}$$

$$q(a_1 \cdots a_m) = \begin{cases} a_1 \cdots a_{n-1} \star a_{m-n+2} \cdots a_m & \text{if } \ m > 2(n-1) \\ a_1 \cdots a_m & \text{else} \end{cases}$$

$$p(a_1 \cdots a_m) = \prod_{g \le i \le m, \star \notin a_{i-n+1} \cdots a_{i-1}} P_{LM}(a_i | a_{i-n+1} \cdots a_{i-1})$$

$$\text{Goal item: } \left[S, 0, |f|, \langle s \rangle^{n-1} \star \langle \backslash s \rangle\right]$$

Figure 2.4: CYK parsing with an integrated n-gram LM [Chiang, 2007]. The inference rules are explored in ascending order of $j-i$. $\alpha\,[e/Y]$ is the string α where the NT occurrence Y is replaced by e. Function q takes as input a sequence of target words and elides words, replacing them with $\star$, that do not impact future n-gram LM calculations, p calculates n-gram LM probabilities for target word sequences that include the $\star$ symbol. $\langle s \rangle^{n-1}$ repeats the $\langle s \rangle$ symbol $n-1$ times.

Consequent chart items maintain *backpointers* to the antecedent items that were used to create them. These backpointers can be used to follow the choice of rules (derivation) that produced the goal item. While the decision rule in Equation 2.9 searches for the highest weight derivation in $\mathcal{D}(G)$, in practice, items that participate in lower weighted derivations are often maintained in the chart in order to generate alternative derivations for discriminative training of weights λ via Minimum Error Rate Training (MERT) [Och, 2003].

This *packed forest* of alternative derivations of a sentence has a hypergraph structure [Gallo et al., 1993, Klein and Manning, 2001]. Hypergraphs are a generalization of directed graphs to allow edges to point to multiple nodes. In Klein and Manning [2001] they define: A directed hypergraph $\mathcal{G}$ is a pair (N, A) where N is a set of nodes and A is a set of directed *hyperedges*. A hyperedge is a pair (T, H) that connects a set of head nodes H and tail nodes T, both of which are subsets of N. The output of a decoder that retains lower weight derivations forms a hypergraph structure where each node in N is a chart item, each hyperedge in A is a PSCFG rule, and the set of tail nodes for a hyperedge is the set of antecedent items used to form the node. The tail nodes on the hyperedge are ordered in left-to-right target order as specified by the rule's target side α.

Figure 2.5 shows an hypergraph formed using the example rules from Section 2.5. Based on this structure, there are efficient algorithms to select the best K weighted derivations [Huang and Chiang, 2005].

2.7 Conclusions

In this chapter we have provided the background for our contributions. We motivated the PSCFG approach based on challenges faced by the traditional word and phrase-

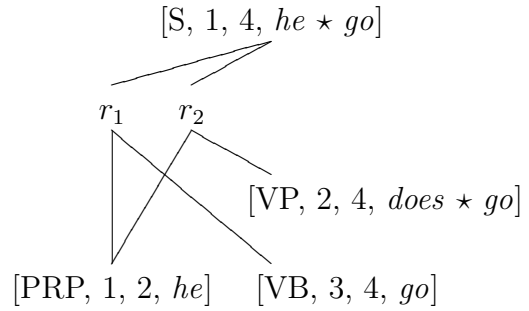


Figure 2.5: A partial hypergraph generated when translating our example sentence *il ne va pas* using the example rules from Section 2.5. Initial rules produce chart items [PRP, 1, 2, *he*], [VB, 3, 4, *go*], [VP, 2, 4, *does* $\star$ *go*], and rules r_1 and r_2 form hyperedges that both produce [S, 1, 4, *he* $\star$ *go*]. $\tilde{e}$ are computed using a q function that represents a 2-gram LM.

based models. We described the rule extraction procedures of the Hiero [Chiang, 2005] and SAMT [Zollmann and Venugopal, 2006] grammars that we will use in this thesis. In Section 2.6 we discussed the bottom-up chart parsing algorithm and structures that we will modify in Chapters 3,5 to make better use of the non-local n-gram LM feature p_{LM} and label constraint p_{syn}.

CHAPTER 3

Two-Pass Approaches to PSCFG Decoding: Hypergraph Search

This chapter develops two-pass approaches to PSCFG decoding based on the intuition that search errors made in an approximate, and therefore fast, first pass can be corrected by performing a second pass search of the first pass hypergraph. Our techniques are focused on efficiently integrating the n-gram feature into PSCFG search for the SAMT grammar. We propose two approaches: a *left-to-right* and *top-down* search of the hypergraph, each corresponding to a different approximation of the p_{LM} feature during first pass decoding. Much of this chapter is based on Venugopal et al. [2007].

3.1 Motivation

As discussed in Section 2.6, new chart items are produced by combining existing chart items with rules to form chart items that represent larger spanning derivations. For each rule: $Z \rightarrow \langle f_{i+1}^{i_1} X_1 f_{j_1+1}^{i_2} Y_2 f_{j_2+1}^{j}, \cdot \rangle$, we need to consider $|\,[X, i_1, j_1, \cdot]\,| \times |\,[Y, i_2, j_2, \cdot]\,|$ possible combinations of existing chart items, where $|\,[X, i_1, j_1, \cdot]\,|$ refers to number of chart items that share the same X, i, j across alternative LM contexts $\tilde{e}$, as indicated by the $\cdot$. Under the MAP decision rule, we are searching for a single sentence spanning derivation; most of the items generated during decoding will not

be used in the highest weighted derivation due to grammar constraints and pruning. Limiting the size of $|\,[X, i_1, j_1, \cdot]\,|$ results in search errors when a discarded item could have combined with other items to create the highest weighted derivation.

In this chapter we explore algorithms that make strong approximations during a *first pass* search, generating a hypergraph that includes derivations that potentially have higher weight than the derivation found in the first pass. In a *second pass*, we explore this hypergraph. Each derivation in this hypergraph is known to be a valid sentence spanning derivation. We propose two methods to search this hypergraph, corresponding to the choice of approximations made during the first pass search. In our first algorithm, called *Left-to-Right* hypergraph search, we remove $\tilde{e}$ from the chart item structure, but continue to use $\tilde{e}$ in order to score consequent derivations. The feature p_{LM} contributes to the choice of first pass derivation but search errors are made. In a second pass search, we generate alternative derivations from the hypergraph in a process that allows the n-gram LM to influence the search. In our second algorithm, called *Top-Down* hypergraph search, we approximate p_{LM} by using an item structure that represents a $g < n$-gram LM context in the first pass. Fewer chart items are produced during first pass decoding at the cost of increased search errors. In a second pass hypergraph search, we adapt the K-Best extraction algorithm from Huang and Chiang [2005] to find better derivations based on the full order n-gram LM.

3.2 Related Work

A baseline approach to the integration of non-local features, like the n-gram LM, is to use these features in a second pass re-ranking of the K-best derivations from the hypergraph. In the field of monolingual parsing, re-ranking has been a successful approach to integrate large number of non-local features [Charniak and Johnson,

2005, Collins and Koo, 2005]. In SMT however, even large K-best lists expose a very small fraction of the search space licensed by the grammar, providing limited potential for non-local features to select better translations. Och et al. [2004] re-rank K-best translation derivations using a large number of features, including more powerful language models and syntactic features. In their experiments, only a single feature, the IBM Model 1 lexical weight, resulted in statistically significant improvements in translation quality.

Using the n-gram LM in a purely re-ranking capacity also yields limited improvements in translation quality [Chiang, 2007, Zollmann and Venugopal, 2006]. The n-gram has a strong impact on translation quality [Brants et al., 2007], making it more important to allow this feature to have a role during search.

Chiang [2007] proposes "Cube Pruning", a single pass approach where the generation of new items is limited during decoding. By using the p_{LM} feature to influence which new items are generated, this approach avoids generating many low weighted items. The Cube Pruning algorithm, which we will compare our algorithms against, is described below.

3.2.1 Cube Pruning: A Single Pass Baseline

Cube Pruning [Chiang, 2007] is an optimization to the intersected LM parsing algorithm presented in Figure 2.4. It addresses the creation of the $|\,[X, i_1, j_1, \cdot]\,| \times |\,[Y, i_2, j_2, \cdot]\,|$ chart items when generating consequent items. Rather than generating all possible consequent items that have item structure $[X, i, j, \cdot]$, Cube Pruning generates only a subset of them.

The search for the approximate best subset of items is performed using the K-best derivation extraction algorithm from Huang and Chiang [2005]. We sketch the Cube Pruning algorithm in Figure 3.1. The algorithm assumes that prior to generating

consequent items $[X, i, j, \cdot]$, there exists a list of rules $\vec{r}$, sorted in decreasing weight w that produce the same lhs=X. The set of possible antecedent items for each nonterminal in r_i is sorted in decreasing weight as well. Rule weights do not include the feature p_{LM}, while antecedent item weights do include p_{LM}. Assume that each rule in $\vec{r}$ has two nonterminal symbols. We represent consequent items as a tuple over antecedent inputs: $\vec{r}(\langle a, b, c\rangle)$ selects the a'th rule from $\vec{r}$, the b'th item from the sorted list of antecedents at the first nonterminal of r_a, and the c'th item at the second nonterminal.

The Cube Pruning algorithm uses the following functions and data structures.

- SCORE($\vec{r}(\langle a, b, c\rangle)$) scores consequent items according to $p(d)$ (Equation 2.16).
- GENERATENEIGHBORS($\vec{r}(\langle a, b, c\rangle)$) generates "neighboring" [Huang and Chiang, 2005] tuples, where each tuple differs from the input argument in one dimension, and calls SCORE on each one.
- h is a priority queue that maintains tuples with weights.
- L is the list of consequent items generated.

CUBEPRUNING creates the consequent chart items L based on Equation 2.16 of Figure 2.4. The functions POPBEST and ADDTOLIST are utility functions to manipulate data structures h and L. Since the function SCORE uses the feature p_{LM}, the n-gram LM influences the set of L generated consequent items. The object *stats* maintains the weight of the best generated consequent item and tracks the number of items generated. CUBEPRUNING terminates when stopping criteria have been reached, typically the number of generated items or the cost difference between the most recently generated item and the best generated item. The termination condition can be relaxed to generate additional lower weighted items to compensate for search

```
CubePruning(r⃗)
 1  AddToH(h, r⃗(⟨1,1,1⟩), Score(r⃗(⟨1,1,1⟩)))
 2  count ← 0
 3  L ← ∅
 4  while !Stop(stats)
 5        do
 6           z ← PopBest(h)
 7           AddToList(L, z)
 8           for z′ ∈ GenerateNeighbors(z)
 9                do AddToH(h, z′, Score(z′))
10           stats ← {cost(z), ++ count}
11  return L
```

Figure 3.1: Generation of a list of consequent items L using the Cube Pruning algorithm [Chiang, 2007]. See description of variables and functions in the text.

error by a "fuzz" factor. In Chiang [2007], Cube Pruning is compared against a) rescoring with p_{LM} only and b) performing traditional beam search with no early stopping during generation of consequent items. Cube Pruning outperforms both the baseline solutions. We will compare our algorithms against Cube Pruning as a strong single pass baseline.

3.2.2 Related Multi-Pass Approaches

Charniak et al. [2006] propose a multi-pass approach to mitigate the impact of nonterminal labels during decoding for the monolingual parsing task. The use of a large number of nonterminal labels in the grammar fragments the search space and often requires pruning to perform search in reasonable times — the authors report a baseline per sentence parsing time of 1 sentence per second. They propose a multi-pass approach where early decoding passes use a reduced nonterminal set. The chart from early decoding passes are used to constrain search in subsequent passes. Zhang and Gildea [2008] propose a multi-pass solution where a lower order n-gram LM is used in the inital pass to constrain the search space in a future pass. This approach is similar to the *Top-Down* approach discussed in Section 3.4, but uses inside and outside heuristics to prune derivations within a full $O(|f|^3)$ second pass decoding, as opposed to using a larger order LM during hypergraph search as we do. Petrov et al. [2008] take a more dramatic step towards reducing the number of chart items generated during decoding. Rather than reducing the order of the n-gram LM, they reduce the target language vocabulary, creating very weak (1-bit and 2-bit) LMs. These weak LMs are used during initial decoding passes and posterior probabilities calculated to estimate the probability of using a particular span during translation. These posterior estimates restrict the possible derivations in subsequent passes that use stronger LMs.

3.3 Two Pass Approaches: Left-to-Right Hypergraph Search

In our first algorithm, called *Left-to-Right* hypergraph search, we remove the LM context $\tilde{e}$ from the chart item structure, but continue to use $\tilde{e}$ that is associated with each derivation in order to score consequent derivations. The feature p_{LM} contributes to the choice of first pass derivation but search errors are made. In a second pass search, we generate alternative derivations from the hypergraph, in a way that provides the left context for the p_{LM} feature.

3.3.1 First Pass: Approximate Parsing

We begin by relaxing the criterion that determines when two chart items are equivalent during parsing. Rather that maintaining $\tilde{e}$ in the chart item, we treat $\tilde{e}$ like α, as a term that is *associated* with each item rather than as an identifying component. In Equation 3.1 we show this approximation explicitly:

$$\frac{Z \rightarrow \langle f_{i+1}^{i_1} X_1 f_{j_1+1}^{i_2} Y_2 f_{j_2+1}^{j}, \alpha \rangle : w \ , \ [X, i_1, j_1], \tilde{e}_1 : w_1 \ , \ [Y, i_2, j_2], \tilde{e}_2 : w_2}{[Z, i, j], q(\alpha') : w w_1 w_2 p(\alpha') \quad (\text{where } \alpha' = \alpha\,[\tilde{e}_1/X_1, \ldots, \tilde{e}_2/X_2])} \tag{3.1}$$

Note that the weight of the consequent item is still calculated based on the $p(\alpha')$, which uses $\tilde{e}_1, \tilde{e}_2$. The runtime complexity of the decoding algorithm is now $\mathcal{O}\left(n^3 |\mathcal{N}|^2\right)$ at the risk of increased search errors. By removing e from the item structure, we commit search errors because derivations with different LM contexts result in the same chart item. This is a form of *greedy* approximation; the item's weight, and associated LM context, represents the derivation with the highest weight among all derivations that share X, i, j, but it is possible that a lower weighted derivation might have composed better with a longer spanning rule due to its e component. This approximation corresponds to an extreme parameterization of Cube Pruning where only one consequent item is propagated for each X, i, j.

This relaxation is different from approaches that do not use the LM during parsing. The derivation weights in the hypergraph *have* LM probabilities factored into them, but represent a non-optimal search according to the model. The hypergraph does, however, represent a set of derivations that successfully translate the source sentence, according to grammar constraints and pruning based on all features in $p(d)$. The second pass of the *Left-to-Right* hypergraph algorithm uses this hypergraph as a starting point to find derivations that potentially have higher weight. New derivations can be found by exploring the alternative hyperedges at each hypernode.

3.3.2 Second Pass: Hypergraph Search

The goal item of the first pass decoding is a chart item that represents a sentence spanning hypergraph of alternative derivations. Instantiating alternative derivations from this hypergraph might yield derivations of lower weight than the first pass maximum weight derivation. This is possible because considering alternative hyperedges that were not selected during the first pass might score better according to p_{LM}.

Exploring the whole space of alternative derivations in this hypergraph is not tractable since there are exponentially many derivations with respect to source sentence length. Selecting the top K derivations and rescoring them based on the LM feature does not yield sufficient diversity to correct search errors made in the first pass [Zollmann and Venugopal, 2006, Chiang, 2007].

We propose a search strategy that explores this hypergraph allowing the p_{LM} feature to influence the sections of the hypergraph that are explored. If we treat the top K derivations from the first pass hypergraph as sample derivations of reasonable quality, then if we make small changes to these derivations, we might find *better* derivations. If a modified derivation has lower weight then we should explore "neighboring" derivations that differ only slightly in their choice of hyperedges.

We define an *agenda item* : $[\tilde{e}, \rho] : w$ based on the components of a hyperedge. Assume a hyperedge representing a rule $r : X \rightarrow \langle \alpha, \gamma \rangle$ with nonterminals X, Y in α and a list of backpointers to chart items: $\vec{b} = [X, i_1, j_1], \tilde{e}_1 : w_1 \ , \ [Y, i_2, j_2], \tilde{e}_2 : w_2$. We define $\rho' = \alpha\,[b_1/X_1, b_2/Y_2]$ where nonterminal elements in α are substituted with pointers to their corresponding antecedent items $b_1, b_2 \in \vec{b}$ represented as hypernodes. ω is defined to be the series of leading consecutive terminal symbols in ρ' while ρ represents the remaining pointers and terminal symbols, therefore $\omega \in \mathcal{T}*$, $\rho \in (\mathcal{T} \cup \vec{b})*$. The weight of the agenda item w is the weight of the derivation that it represents. Item component $\tilde{e} = q'(\omega)$ where $q'(\omega)$ is defined similarly to the q function in Figure 2.4 as:

$$q'(\omega_1 \cdots \omega_m) = \begin{cases} \omega_{m-n+2} \cdots \omega_m & \text{if } \ m > n-1 \\ \omega_1 \cdots \omega_m & \text{else} \end{cases}$$

Unlike q, q' only retains the right most n-1 words from ω.

We conduct our second pass search through the hypergraph via agenda items. Agenda items are sorted by weight in a map data structure B, which maintains only unique agenda items and discards duplicate items of lower weight. B is initialized with agenda items corresponding to the hyperedges of the sentence spanning goal item. Each of these agenda items represents a section of the hypergraph to be explored. To conduct our search for alternative derivations we explore "neighboring" agenda items using the function UNWIND in Figure 3.2.

The UNWIND function takes the agenda B and an agenda item x as input, where x has already been removed from B. x is a hypergraph where some of its left most branches have been instantiated into specific derivations, resulting in a sequence of left most target terminal symbols. UNWIND generates new agenda items that represent alternative derivation choices at the remaining left most branches of the hypergraph.

```
UNWIND(B, β_τ, x = [ẽ, ρ] : w)
 1  bp_1 ← GETFIRSTBP(ρ)
 2  [ẽ_1, ρ_1] : w_1 ← AGENDAITEM(bp_0, 1)
 3  ρ', i_1, j_1 ← EXPAND(ρ, bp_1, [e_1, ρ_1])
 4  w' ← FACTOROUT(w, ẽ, ρ', i_1, j_1, w_1)
 5  for 1 ≤ k ≤ |edges(bp_1)|
 6        do
 7           [ẽ_k, ρ_k] : w_k ← AGENDAITEM(bp_1, k)
 8           ρ'_k, i_k, j_k ← REPLACE(ρ', i_1, j_1, [ẽ_k, ρ_k])
 9           w_k ← FACTORIN(w', ẽ, ρ'_k, i_k, j_k, w_k)
10           ẽ_k, ρ_k ← q'(ẽ · PREFIX(ρ'_k)), SUFFIX(ρ'_k)
11           if BESTW(B)/w_k ≤ β_k
12              then ADDTOB(B, [ẽ_k, ρ_k] : w_k)
```

Figure 3.2: UNWIND(B, x) generates new agenda items based on x that differ in their choice of hyperedge at x's left-most backpointer. Generating new agenda items in this way produces sequences of consecutive terminal symbols in ρ.

By generating alternatives in this way, the p_{LM} feature can be used to score consecutive terminal sequences that start from the beginning of the sentence, unlike in first pass decoding. When new agenda items are created, they can be efficiently scored with the p_{LM} feature. New items that have high weight should be explored further, while items that have a low weight relative to the best item found should not be explored further.

We describe each subroutine in UNWIND below:

- GETFIRSTBP: Returns the left-most backpointer to a chart item from ρ, which is a sequence of target terminals and backpointers.

- AGENDAITEM: Takes a backpointer to a chart item and a hyperedge index and forms an agenda item as described above. This function assumes that the hyperedges in the chart item are sorted by weight where 1 represents the hyperedge with highest weight.

- EXPAND: Takes an existing ρ and replaces the first backpointer with its corresponding agenda item. The result is an expanded sequence ρ'. We also track the positions in ρ' that represent this substitution. ρ' now represents a hypergraph where one of the nodes in ρ has been instantiated as a (non-hyper) edge.

- FACTOROUT: Removes the weight contributions from w that came from $[e_1, \rho_1] : w_1$ during first pass parsing. The weight contributions include the weight from the p_{LM} features as well as other feature functions h_i. The left context $\tilde{e}$ allows the p_{LM} feature to be calculated for the sequence $\tilde{e}, \rho'$.

- REPLACE: Replaces elements in ρ' that correspond to the original edge, represented by symbols $\rho'_{i_1} \cdots \rho'_{j_1}$, with $\tilde{e}_k, \rho_k$—an alternative agenda item. This represents following an alternative hyperedge at the left-most backpointer of x.

Returns a new sequence ρ' with positions indicating the relative location of the new agenda item.

- FACTORIN: Multiplies in the weight contribution due to selecting hyperedge k. The weight contributions include the weight from the p_{LM} features as well as other feature functions h_i. Using left target terminal context $\tilde{e}$ allows the p_{LM} feature to be calculated for the sequence $\tilde{e}\rho'_k$. The resulting w_k can be higher than w, representing a recovery of search error.

- PREFIX: Returns the leading consecutive terminals in ρ'_k.

- SUFFIX: Returns all symbols after and including the first backpointer in ρ'_k.

- BESTW, WORSTW: Returns the highest and lowest weight of agenda items that have *no backpointers* in B. If all the elements in B still have backpointers, then these functions return values that indicate this condition.

- ADDTOB: Adds a new agenda item to B. If the agenda already has an item with the same structure, the item with highest weight is retained.

The FACTORIN and FACTOROUT functions call the p function to calculate the LM probability of a sequence $\tilde{e}, \rho$. Before calling the p function, backpointers items in ρ are substituted with their corresponding $\tilde{e}$.

The LEFTTORIGHTSEARCH search algorithm, defined in Figure 3.3, uses UNWIND to generate alternative chart items in sections of the hypergraph that have high weight. At each iteration during search, every item that still has backpointers in the agenda is explored by UNWIND. Newly generated items with low weight, relative to the best agenda item with no backpointers, are discarded, while higher weighted items remain in the beam and are explored further. Search is terminated by considering the items on the agenda that have no backpointers. When the difference in weights

for these items is greater than β_τ, search is terminated. We now highlight particular characteristics of this search algorithm.

The UNWIND function generates new agenda items in left-to-right order, maintaining a sequence of left-most consecutive target terminal symbols. This search strategy is similar in motivation to the grammar constraint applied in Watanabe et al. [2006]. They propose restricting the PSCFG grammar to include only rules that have Griebach Normal form, so that all target terminal symbols in γ come before nonterminal symbols. This reduces the number of chart items created during decoding since all items will share the same left context.

The agenda item structure allows us to discard items under the MAP decision rule just as we did during first pass search. Duplicate agenda items with lower weight are discarded. Matching of agenda items $[\tilde{e}, \rho] : w$ is performed by hashing the target terminals in $\tilde{e}, \rho$ and memory addresses of the backpointers in ρ. In order to perform discriminative training we need to be able to generate K-best alternative derivations. By maintaining backpointers to duplicate items with lower weight we can run the algorithm from Huang and Chiang [2005] to extract alternative derivations from the hypergraph of agenda items.

The computationally expensive portion of the Left-to-Right hypergraph search algorithm is the generation of new agenda items. For each agenda item we need to call FACTOROUT once, and FACTORIN once for each alternative that is generated from it. These functions need to recalculate the p_{LM} feature based on agenda item components $\tilde{e}, \rho$. Rather than calculating n-gram language model probabilities for all symbols in $\tilde{e}, \rho$, we use the position information i_k, j_k to limit the number of new probabilities requested. We only need to request new probabilities at the symbols that surround the i_k, j_k.

Second Pass Example: Consider a simple example, where we have completed

```
LeftToRightSearch(g, β_τ)
 1  B ← ∅
 2  for 1 ≤ i ≤ |edges(g)| AddToB(GetAgendaItem(g, i))
 3  while !Stop(BestW(B)/WorstW(B) > β_τ)
 4        do
 5           B' ← ∅
 6           while |B| > 0
 7                 do
 8                    a ← Pop(B)
 9                    if |bps(a)| > 0
10                       then Unwind(B', a)
11                       else  AddToB(B', a)
12           B = B'
```

Figure 3.3: LeftToRightSearch(g) takes as input the goal item g and a pruning parameter β_τ. Agenda items are formed based on the hyperedges of the goal items and alternative agenda items are explored using the Unwind function. The search maintains a sorted agenda B where unique agenda items are maintained. Search terminates when the weight ratio of the best and worst agenda item's that have no nonterminals is greater than β_τ.

the first pass search and produced a sentence spanning goal item. Figure 3.4 shows the goal item label S.

This goal item has two hyperedges, each representing an alternative reordering of antecedent chart items. The solid arrow represents the hyperedge that was selected during first pass decoding and the dashed arrow represents an edge that was not selected.

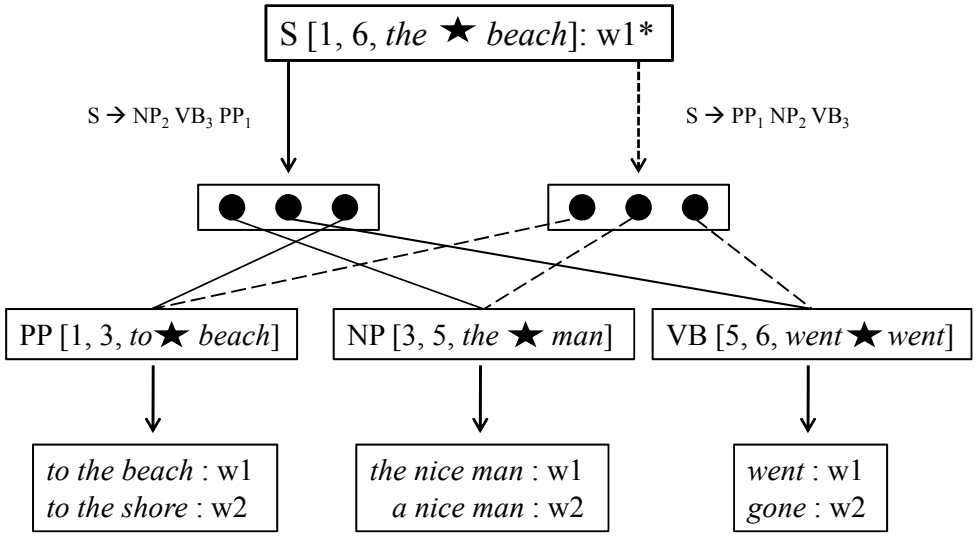
Each hyperedge has a tail of backpointers to chart items that represent subspan translations. In this example, the hyperedges point back to PP, NP and VB chart items [1].

At each chart item, we have listed the n-gram LM context assuming a 2-gram LM, with one word at the left boundary and one word at the right boundary, from the first pass search. The antecedent chart items represent multiple purely lexical rules (no nonterminal symbols) in the boxed regions below them.

Each iteration of the agenda B is indicated by a boxed list. The first list includes two agenda items representing the two alternative hyperedges that lead into the goal item. In this example, $w1*$ indicates the weight of the derivation from first pass decoding. In the second iteration, we UNWIND both agenda items. The initial item $[\tilde{e} = \emptyset, \rho = \text{NP}_2\ \text{VP}_3\ \text{PP}_1]$ generates two new agenda items that have the same item structure: $[\tilde{e} = man, \rho = \text{VP}_3\ \text{PP}_1]$; only the item with higher weight is retained. We maintain a pointer to the lower weight item, as indicated by the arrow in the example. The last agenda item in the second iteration list is an example of an agenda item that has been pruned due to its relative performance, as indicated by the strike through. As per our algorithm, we explore alternative items for each element in B during the third iteration.

[1] In our experiments, we limit SAMT rules to have at most two nonterminal symbols, but for the purposes of illustrating our second-pass algorithm, we use rules with three nonterminal symbols.

Hypergraph after first pass



Agenda B

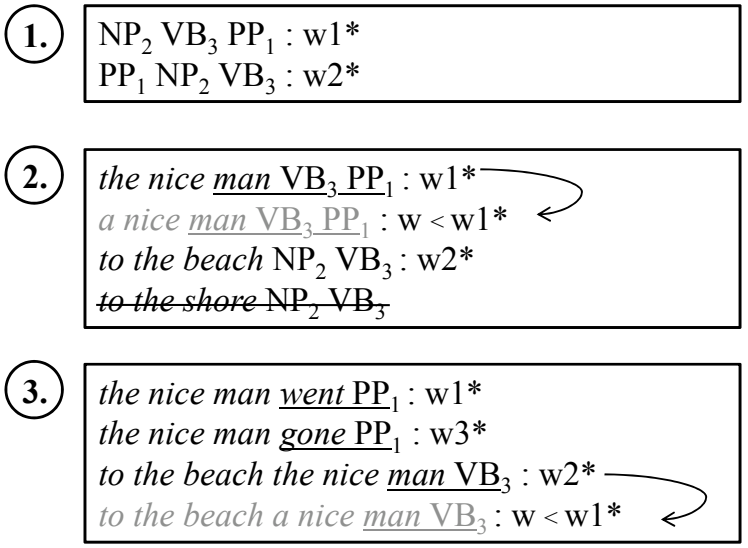


Figure 3.4: Illustration of the second-pass of the Left-to-Right search algorithm. The result of the first pass is the hypergraph rooted at goal item item S. The hypergraph structure and agenda items are explained in the text.

3.3.3 Empirical Results: Left-to-Right Search vs. Cube Pruning

We present empirical results on the IWSLT 2006 Chinese to English translation task [Paul, 2006]. Training and development data are described in Appendix A, Table A.1. A 5-gram LM was trained based on the target side of the parallel corpus using Kneser-Ney smoothing [Kneser and Ney, 1995].

We compare the single pass Cube Pruning (CP) algorithm [Chiang, 2007] (described in Section 3.2.1), with the two pass Left-to-Right (LR) hypergraph search algorithm from Section 3.3. We do not evaluate our approach against the simpler rescoring and full search with beam pruning baseline options. This comparison is available in Chiang [2007] where they show that CP outperforms both of these alternatives.

We evaluate the Left-to-Right algorithm using the Hiero ([Chiang, 2005], Section 2.4) and SAMT ([Zollmann and Venugopal, 2006], Section 2.5) grammars. The SAMT grammar rules use labels from target language parse trees generated by the Stanford Parser [Klein and Manning, 2003]. The Hiero grammar contains 50K rules, while the SAMT grammar contains 300K rules. Parameters λ are trained using MERT training [Och, 2003] on the development set IWSLT DevSet4 (Table A.1) using the baseline CP decoding algorithm.

We evaluate each decoding algorithm by considering search errors made during decoding and by the BLEU score on evaluation data. We define search errors based on the weight assigned by the model to the goal item for each sentence in the evaluation corpus. Treating weights as negative log probabilities (costs), we accumulate the value of these *model costs* for each sentence in the evaluation corpus as we vary beam pruning settings.

We vary β_w (which specifies the maximum allowed weight difference between chart

items that share X, i, j) for CP, and β_τ in the second pass of LR. β_w for the first pass in LR search is 5. β_τ, β_w are given as multiples of the p_{LM} feature weight in λ. Large values of β_τ, β_w correspond to exploring more alternative derivations. In the experiments below we also limit the size of B to 1000 for memory considerations.

The first pass of LR can be performed with no pruning. If the accumulated cost is reduced using wider beam settings, search errors have been reduced over previous beam settings. For CP we accumulate the weight of the goal item after the single search pass and for LR we accumulate the weight from BestW after the second pass. We compare decoding algorithms by evaluating the accumulated model cost versus *effort* made during search.

We consider two measures of effort. We first evaluate search errors as a function of novel queries made to the n-gram LM. It is our intuition that novel queries to the n-gram LM represent areas of the search space that have not been previously explored. Ideally, a decoding algorithm will quickly explore only the most promising regions of the search space and result in low accumulated model cost. We propose this measure in order to abstract away some of the low level implementation decisions that apply to each algorithm. Novel queries are also an important decoding statistic when using distributed LMs like those in Brants et al. [2007]. Repeated n-gram queries can be maintained efficiently in local caches while novel queries require time consuming remote procedure calls. When we compare algorithms under this measure, the better algorithm would achieve each level of search error with fewer novel queries to the n-gram LM.

The second measure is the wall-clock time required to achieve a particular level of search error; the better algorithm would achieve each level of search error in less time. In order to demonstrate the relationship between the BLEU score and search error, we report BLEU score results at each level of pruning. We use wider beams for

both algorithms until we see no change in BLEU score.

Figures 3.5,3.6 plot model cost as a function of LM cache misses for the Hiero and SAMT grammars, while Figure 3.7 plots model cost as a function of decoding time for the SAMT grammar.

For both the Hiero and SAMT grammars we see that LR achieves a given model cost earlier in terms of novel LM calls for most of the plotted region, but ultimately fails to achieve the same lowest model cost as the CP method. Under both LR and CP, the Hiero grammar achieves a BLEU score of 19.1%, while the SAMT grammar's score is 1.6% higher at 20.7%. Hiero demonstrates a greater variance of BLEU score for both CP and LR compared to the SAMT grammar. The use of syntactic labels as an additional model of target language fluency might explain the fact that the SAMT grammar quality is more robust to differences in the number of items explored that differ in their n-gram LM context. We see similar results when considering decoding time for the SAMT grammar. While LR achieves a relatively low model cost quickly, it ultimately does not achieve the same low model cost as CP.

The Left-to-Right search algorithm demonstrates that it is possible to make strong approximations during first pass decoding and partially recover from them during a second pass. Ultimately, Left-to-Right search does not outperform the strong single pass Cube Pruning baseline, failing to achieve equally low model costs. The second pass of LR search uses derivation weights from first pass decoding to guide the search for alternative derivations in the hypergraph. Methods to underestimate the cost of agenda items that still have unexplored backpointers could lead to a better search heuristic (ideally an admissible heuristic [Dechter and Pearl, 1985]) that can guide the second pass.

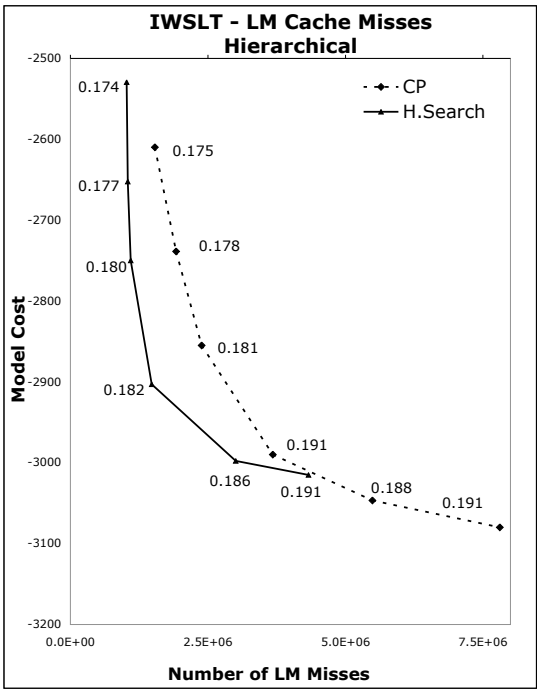


Figure 3.5: Model cost vs. LM cache misses with BLEU scores for the IWSLT Hiero grammar for Cube Pruning and Left-to-Right hypergraph search.

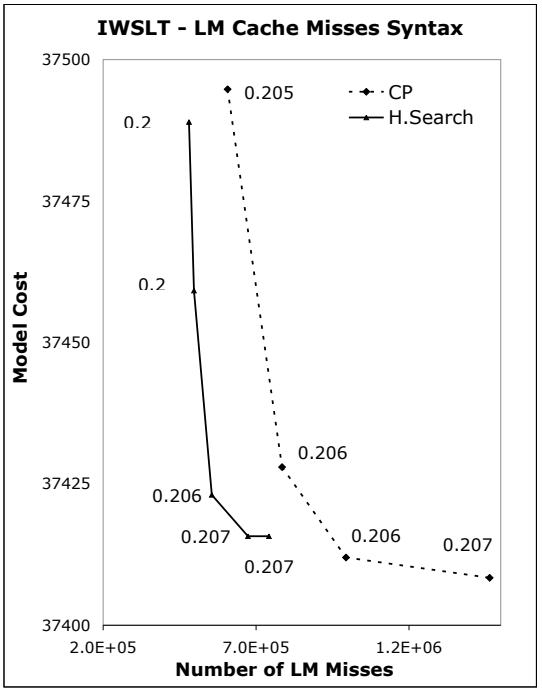


Figure 3.6: Model cost vs. LM cache misses with BLEU scores for the IWSLT SAMT grammar for Cube Pruning and Left-to-Right hypergraph search.

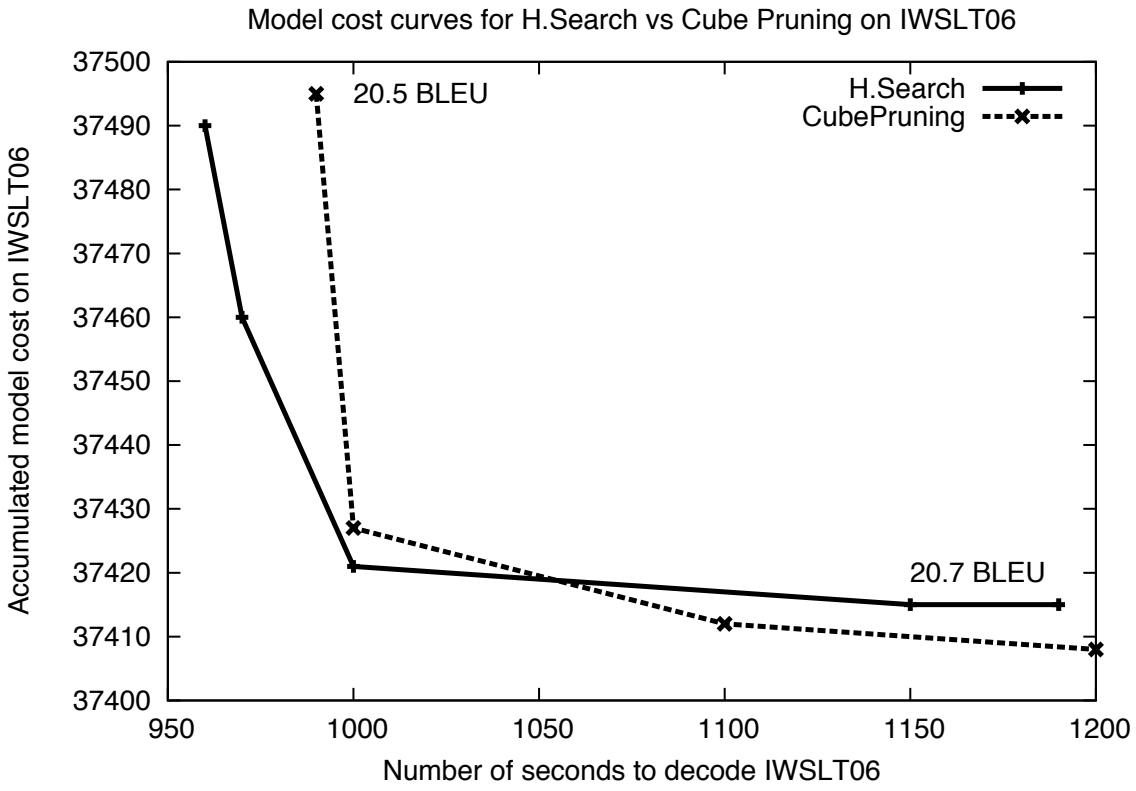


Figure 3.7: Model cost vs. decoding time for the IWSLT SAMT grammar comparing Cube Pruning and Left-to-Right hypergraph search.

3.4 Two Pass Approaches: Top-Down Hypergraph Search

In the Left-to-Right approach described above, we reduced the search space in the first pass by only the preserving best chart item amongst those items that shared X, i, j, but retained the lower weighted alternatives for a second pass hypergraph search. The implied approximation was that local greedy decisions during the first pass would generate a hypergraph representing high scoring derivations that could be further explored in a second pass. The search strategy in the second pass provided additional context (leading target terminals) for the p_{LM} feature to drive this search.

In this section we explore an alternative approximation. Rather than making greedy decisions about the number of chart items to propagate, we make an approximation in the context that identifies derivations in the item structure. Rather than maintaining sufficient LM context in the item to make locally optimal decisions according to the n-gram LM (the operation performed by the q function in Figure 2.4), we maintain LM context in the item as if we were using a lower order $g < n$-gram LM. This causes non-optimal decisions to be made during first pass search, with respect to the full n-gram model used by the p_{LM} feature. In a second pass, we explore the first pass hypergraph in a *Top-Down* manner, similar to the K-best algorithm of Huang and Chiang [2005], to select derivations according to the full order n-gram LM.

3.4.1 First Pass: Approximate Parsing

In first pass decoding, we follow the bottom-up decoding algorithm from Figure 2.4 with the following modifications to the q function. Rather than using $n-1$ words to maintain LM context, we use $g-1$ words, where $g < n$:

$$q(a_1 \cdots a_m) = \begin{cases} a_1 \cdots a_{g-1} \star a_{m-g+2} \cdots a_m & \text{if } \ m > 2(g-1) \\ a_1 \cdots a_m & \text{else} \end{cases}$$

The p function, which calculates the feature p_{LM} still continues to use the n-gram LM:

$$p(a_1 \cdots a_m) = \prod_{g \leq i \leq m, \star \notin a_{i-n+1} \cdots a_{i-1}} P_{LM}(a_i | a_{i-n+1} \cdots a_{i-1})$$

The result of this decoding approximation on runtime is shown below, where $g-1$ is used instead of $n-1$:

$$\mathcal{O}\left(n^3 \left[|\mathcal{N}||\mathcal{T}_T|^{2(g-1)}\right]^2\right)$$

First pass parsing will generate fewer chart items since more derivations will produce the same chart item structure when considering only $2(g-1)$ words of LM context. More search errors are introduced since lower weighted derivations that share the same item structure have the potential to form higher weighted derivations when scored with the complete n-gram LM in p_{LM}.

In the second stage, we attempt to recover from the potential search errors made by the approximation above. As shown below, rescoring a large K-best list extracted from the first stage fails to achieve the lowest levels of search error. As in the Left-to-Right method above, we will use the full-order language model to drive the search through the hypergraph generated in the first pass, but we now use a Top-Down (instead of Left-to-Right) strategy, which is effectively a modification of the K-best extraction algorithm of Huang and Chiang [2005].

3.4.2 Second Pass: Top Down Search

Huang and Chiang [2005] describe an efficient algorithm (Lazy-K-Best) to select the

top K ranked derivations from a hypergraph. The algorithm works recursively, requesting the top ranked derivation corresponding to the goal item in the hypergraph triggers the same request along each incoming hyperedge. At each hyperedge, the request for the first best derivation is passed down to each backpointer item. Requesting the second best derivation follows the same strategy, but takes advantage of previously computed derivations at each chart item. As defined in Huang and Chiang [2005], this algorithm computes K-best derivations in decreasing weight order from a hypergraph. We use this algorithm to correct for search errors made during approximate first pass decoding. Our approach is similar to recent work in Huang and Chiang [2007].

We sketch the recursive component of the algorithm from Huang and Chiang [2005] in Figure 3.8, it is similar to the Cube Pruning algorithm in Figure 3.1, but refers to hypergraph elements that were created during decoding. The input to GetKthBest is a chart item x, and a request for the K-th best derivations at x. Two data structures are maintained at each x; a list of weight derivations rooted at x accessed by $L(x)$, and a priority queue of possible alternative derivations $h(x)$ to explore. Assuming that each rules has two backpointers, alternative derivations are represented as tuples $\langle a, b \rangle$, where a refers to the a'th incoming hyperedge of x, and b corresponds to the choice of the b'th best derivation at the first backpointer of the a'the edge. Each of these derivations is weighted according to the model $p(d)$ by the Score function.

The recursive component of the GetKthBest algorithm is represented by HasKthBestAll and GetKthBestAll. HasKthBestAll checks each input tuple $\langle a, b \cdots \rangle$, following x's a'th edge to each of its backpointers chart items. If the backpointer item has already generated its b'th best derivation and the same is true at all other backpointers, HasKthBestAll returns true. If any of these requested derivations are missing, GetKthBestAll recursively calls GetKthBest on the corresponding

```
GetKthBest(x, K)
 1  h ← ∅
 2  if x(h) == ∅, x(L) == ∅
 3      then
 4           for 1 ≤ i ≤ |edges(x)|
 5                do AddToH(x(⟨, i, 1⟩), Score(x(⟨i, 1⟩)))
 6  if |L| ≥ K
 7      then return L(K)
 8  while |L| < K
 9        do
10           z ← PopBest(h)
11           AddToList(L, z)
12           for z′ ∈ GenerateNeighbors(z)
13                do
14                   if !HasKthBestAll(z′)
15                      then GetKthBestAll(z′)
16                   AddToH(z′, Score(z′))
```

Figure 3.8: The recursive component of the Lazy-K-Best extraction algorithm from Huang and Chiang [2005]. We use this algorithm in a second pass search in the Top-Down algorithm. Function calls and variables are defined in the text.

backpointer items.

SCORE evaluates a choice of hyperedge and derivation choices at each of its backpointers and evaluates this choice based on $p(d)$. When the same order n in the q function, as is used to evaluate the n-gram LM in $p(d)$, the algorithm will always return derivations of strictly increasing weight.

In this *Top-Down* approach, using a restricted LM context during decoding will result in derivations found with lower weight during K-best extraction. This represents a recovery from search errors. K is therefore the parameter that controls our exploration of the hypergraph in the second pass. Note that $L(x)$ is not resorted as new derivations are found, doing so would make the algorithm return different derivations at each choice of K at a particular chart item. Unlike Left-to-Right search, which maintains a single sentence level agenda, Top-Down search effectively maintains an agenda of alternative derivations to explore at each node in the hypergraph.

3.4.3 Empirical Results: Top-Down Search vs. Cube Pruning

Top-Down search is based on the intuition that MAP decisions made based on the lower n-gram LM context serves as a good approximation to use the full order n-gram LM context, and that the resulting search errors can be corrected in a second pass search. We evaluate Top-Down (TD) search against Cube Pruning (CP) by considering accumulated model cost just as we did above on the IWSLT (Appendix A) translation task. Note that Top-Down search still uses CP during first pass search. Results below are presented on the IWSLT 2007 test set, with development parameters trained on IWSLT DevSet4 (Table A.1). The test sets from previous years (2005, 2006) have been added to the training data, resulting in improved BLEU score performance relative to experiments in Section 3.3.3.

We generate model cost versus total decoding time curves for three systems —

CP, TD and rescoring. The CP system uses a 5-gram LM (system "5-gram-CP"), the TD system maintains effective 3-gram context during the first pass followed by the full 5-gram LM in the second pass (system "3-gram-TD") and the rescoring system that uses the 3-gram context during decoding but then rescores a K-best list (system "3-gram-Res").

We use the SAMT grammar for these experiments. Figure 3.9 shows relative performance between the three systems mentioned above, using K=1000 for the system 3-gram-TD, and K=5000 for system 3-gram-Res. The relatively higher K is used for the rescoring system since we want to demonstrate that even with a large K we cannot recover from search errors by K-best list rescoring.

The absolute pruning threshold β_n is varied in these experiments, limiting the number of consequent items propagated during CP. The CP fuzz factor is set to 10000 items. Note that the TD approach uses CP in its first pass, except with the shortened LM context. We see that 3-gram-TD achieves the lowest model cost in 53% less time (36904 seconds for the 5-gram-CP system and 17112 seconds for the 3-gram-TD system) than the 5-gram-CP single pass CP system. 3-gram-TD also achieves the same low model cost achieved by the 5-gram-CP system. The TD approach achieves a lower cost than the rescoring method.

BLEU scores on Figure 3.9 are placed near corresponding pruning levels on each system to show improvements in translation quality when search error is reduced. There is an improvement in the BLEU score from 36.7% to 37.4% when using larger beam values β_n for the rescoring method. However, the additional reduction in accumulated cost from the best rescoring system to the Cube Pruning and TD systems does not result in additional BLEU score improvement, all systems ultimately achieve a score of 37.4%.

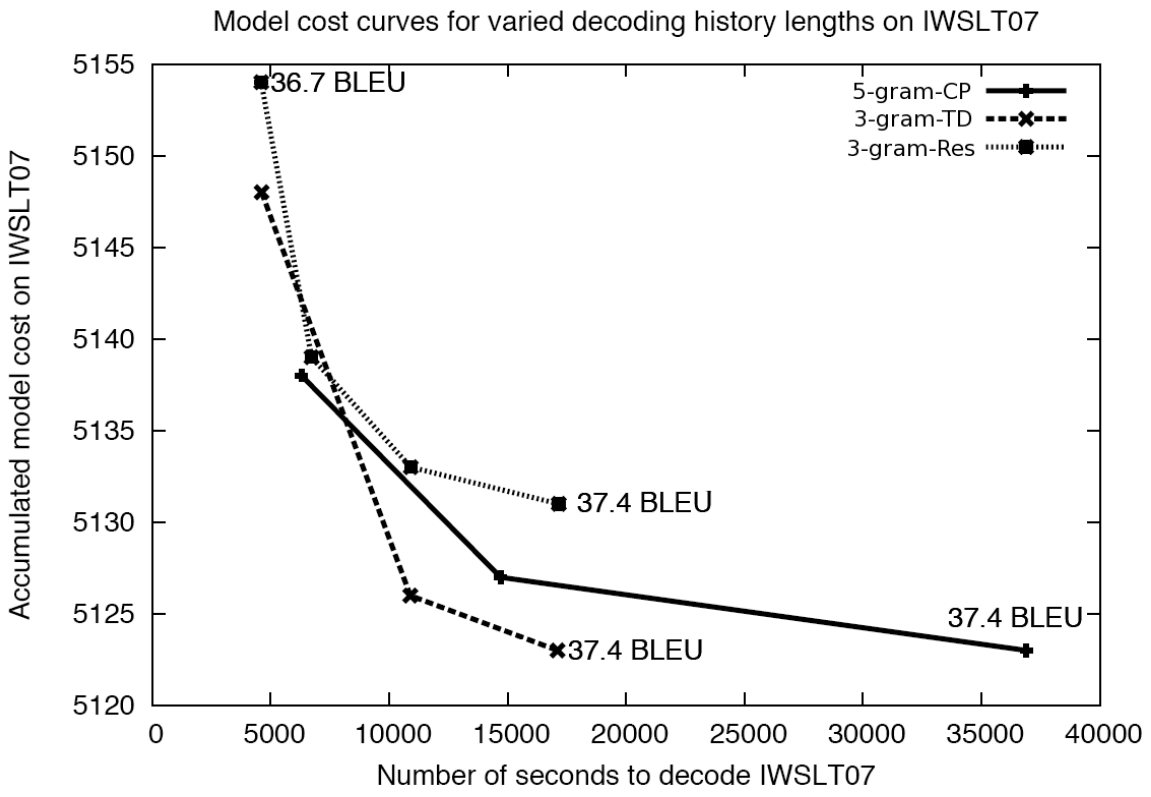


Figure 3.9: Decoding time for IWSLT SAMT grammar and BLEU scores for varied pruning parameters comparing the Top-Down approach (3-gram-TD) to two-pass rescoring (3-gram-Res) and to Cube Pruning (5-gram-CP).

3.5 Conclusions and Contributions

In this Section we focused on the problem of efficiently integrating the n-gram LM feature into a PSCFG decoder. We proposed two-pass approaches to mitigate the increase in the number of chart items during decoding when fully integrating the n-gram LM into search. Our approaches are based on the intuition that search errors made during an approximate, fast first pass can be corrected in a more extensive search of the resulting hypergraph.

The first pass is used to narrow the search space to those derivations that successfully translate the source sentence under grammar and pruning constraints, while the second pass is able score partial derivations by taking advantage of more contextual information. In the Left-to-Right algorithm, the second pass is conducted so consecutive target terminals are produced at each new derivation allowing the n-gram LM to have a complete history when applied in the feature p_{LM}. In the Top-Down algorithm, the second pass rescores partial derivations in the context of higher up rules in the sentence spanning derivation.

We compared the performance of our two-pass approaches against a strong Cube Pruning [Chiang, 2007] baseline, by evaluating the effort each approach made to achieve a particular level of search error. Search error was measured by considering the accumulated model cost of the selected translation in a test corpus. While Left-to-Right search achieves low search error quickly, it is unable to achieve the lowest level of search error that was achieved by the Cube Pruning approach. The Top-Down algorithm achieves the lowest level of search error in our experiments in 53% less time than single pass Cube Pruning for the SAMT grammar. The methods developed to establish improvements in search strategy by measuring model cost were novel as of their presentation in Venugopal et al. [2007], and several new methods use this strategy to demonstrate improvements [Huang and Chiang, 2007, Zhang and Gildea,

2008].

These search strategies can be applied to the broad class of PSCFG grammars that generate synchronous derivations on the source and target language. Beyond n-gram models, our work proposes general strategies to integrate non-local features, taking advantage of knowledge about their contextual requirements and fashioning our search based on this information. In Chapter 5 we use the ideas developed here to integrate syntactic constraints as a non-local feature into the translation model. We expect our approximations to be more valuable when non-local information causes greater increases in the number of chart items during decoding, the feature we explore in Chapter 5 has this effect.

CHAPTER 4

A Systematic Comparison: Phrase-Based vs. PSCFG

As discussed in Chapter 2, PSCFG based approaches offer the potential for translation and reordering decisions to be made with more contextual information than typically available to phrase-based models. In this chapter we perform a systematic comparison between PSCFG approaches (Hiero and SAMT) against a strong phrase-based baseline using the same initial lexical phrases and reordering constraints across all systems, with a focus on understanding the impact of the non-local n-gram LM and syntactic label features. We evaluate these systems across a variety of resource conditions to translate Chinese, Arabic and Urdu to English. Our work shows that PSCFG methods deliver consistent improvements, even when all systems have access to large language models and we show further improvements when using syntactic labels for languages that exhibit long distance reordering, such as Chinese and Urdu. Most of the work in this chapter was published in Zollmann et al. [2008].

4.1 Motivation and Related Work

PSCFG methods introduce extensions beyond the traditional phrase-based systems like Och and Ney [2004], Koehn et al. [2003], Vogel [2005]. These extensions, like the identification of rules with nonterminal symbols and corresponding decoder mod-

ifications, have a significant runtime impact and present challenges when used for large scale tasks. In this chapter, we focus on isolating the relative improvements due the PSCFG grammars when compared against a large scale, strong, phrase-based baseline. We break down this impact into improvements using Hiero and further improvements when using SAMT. We use the phrase-based system from Brants et al. [2007] as a strong baseline to compare performance against the PSCFG methods. We run these comparisons on three standard NIST evaluations tasks, Chinese-to-English, Arabic-to-English and Urdu-to-English,each representing unique challenges in statistical machine translation.

The purpose of this evaluation is to understand the effectiveness of PSCFG approaches. It is important to understand the circumstances in which PSCFG approaches can deliver improvements in translation quality. These circumstances range from the choices of language pair, the size of the available parallel and monolingual training corpora and the use of nonterminal labels in the grammar. Furthermore, given the sensitivity of machine translation systems to the data sets used to evaluate them, we present a comprehensive evaluation on multiple test corpora.

We pay particular attention to the influence of non-local features on translation quality. Significant work has been done to efficiently integrate n-gram LMs into PSCFG decoding [Chiang, 2007, Venugopal et al., 2007], discussed in Chapter 3. We would like to know whether phrase-based systems, when capable of the same long distance reordering as PSCFG systems, can achieve the same performance as PSCFG systems when strong LMs are available. We would also like to know whether the benefits of nonterminal labels persist when used in concert with strong n-gram LMs.

Systematic empirical comparisons have played an important role in the development of machine translation models and systems. Och and Ney [2003] compare alternative alignment models which form the basis for most phrase-based and PSCFG

approaches. The development of multilingual resources like Koehn [2005] have facilitated work like Callison-Burch et al. [2009], where multiple systems and approaches are evaluated on multiple language pairs.

4.2 Experimental Framework

We begin with a description of the experimental framework used for our evaluation. We describe model parameters from each system and discuss additional components that are required to facilitate this comparison.

4.2.1 Phrase-Based Baseline

We compare Hiero and SAMT against a strong phrase-based baseline, which is able to perform the same long distance reordering effects as these PSCFG systems. Results reported in Chiang [2005, 2007], show significant improvements when using a PSCFG grammar, with a single nonterminal symbol, when compared against a phrase-based baseline. The PSCFG grammar in Chiang [2005, 2007] allowed the decoder to reorder phrases within a window of 10 source words, while the phrase-based system was limited to a 4 word window. Similarly, in Marcu et al. [2006], Galley et al. [2006], a PSCFG system where rules are learned based on phrase structure parse trees, shows improvements over a phrase-based baseline that is limited to a reordering window of 7. In this work, our phrase-based baseline uses reordering limits that match the reordering capabilities of the PSCFG grammars, thereby isolating the impact of the nonterminal based approach to reordering. Furthermore, all PSCFG rules are learned from the *same initial phrases* that are used within the phrase-based system. This removes possible variance due to phrase identification techniques.

In our experiments, initial phrases are extracted by the method described in Och

and Ney [2004] and phrase-based reordering during decoding is modeled by the lexicalized distortion approach in Zens and Ney [2006a]. All initial phrase pairs are limited to have source length of at most 6 words and target length of at most 12 words, allowing for significant differences in source and target lengths for each phrase. The reordering limit β_{reo} for the phrase-based system is increased until there are no additional improvements.

The p_{LM} feature is computed using several, large, high order (up to seven gram), distributed n-gram LMs as described in Brants et al. [2007]. These models are able to report to the decoder when an n-gram probability is the result of a backoff event [Kneser and Ney, 1995].

4.2.2 PSCFG Systems

We build systems using the Hiero and SAMT grammars. These systems were chosen for comparison because they differ primarily in their choice of nonterminal labeling strategy. Unlike in Chiang [2005], PSCFG rules that have no remaining alignment points are allowed in our grammar. Loosening this restriction allows both PSCFG grammars to explicitly represent deletion of source words and insertion of target words in the context of nonterminal symbols. Parse trees for the SAMT system are generated on the target side of the bilingual corpus with the stochastic parser from Charniak [2000], which is trained on the Penn Treebank corpus [Marcus et al., 1993].

4.2.3 Minimal State Language Models

The n-gram LMs described in Brants et al. [2007] are able to report when the probability of an n-gram is the result of a backoff event. Under the decoding algorithm from Chiang [2007], chart items are identified by $2(n-1)$ context words, causing an explosion of chart items for high order n-gram LMs. In order to efficiently inte-

grate these large language models into the PSCFG decoder, we take advantage of the reported backoff information.

We note that the full $n-1$ left and right word histories are unnecessary to safely compare two competing chart items when using an n-gram LM. Rather, given the sparsity of high order n-gram LMs, we only need to consider histories that can be *found* in the n-gram LM. This allows more derivations to share the same chart item during decoding, without the risk of additional search error. The n-gram LM implementation, described in Brants et al. [2007], indicates when a particular n-gram is not found in the model and returns a shortened n-gram or *minimal state* to represent the shortened n-gram that was found. This state is then used to identify the left and right chart item histories and differentiate between chart items.

We now describe the modifications required to efficiently use minimal states rather than a fixed $n-1$ words in the chart item structure. We modify the q function, which generates the language model context e in the chart item, to return a context that includes both terminal symbols and minimal states. Minimal states are terminal sequences $[a_1 \cdots a_j]$. The language model is able to return probabilities for terminal sequences, as well as a terminal sequence given a state, i.e $p([a_i \cdots a_j]\ a_{j+1} \cdots a_m)$. The p function as defined in Figure 2.4 does not change, we treat the state markers as annotations on the terminal elements rather than a separate token type.

The input to the q function is a sequence of terminals and minimal states, $a_1 \cdots a_m$. The modified q function as well as additional helper functions are defined in Figure 4.1.

The goal of the q function is to determine an identifying left and right language model context $\tilde{e}$ based on $a_1 \cdots a_m$ to allow subsequent language model calculations to be performed exactly. The context consists of one or two parts, separated by the $\star$ symbol as previously defined in Figure 2.4, except now, the right context is represented by a minimal state. We need to retain the first j terminals of $a_1 \cdots a_m$,

$$q(a_1 \cdots a_m) = \begin{cases} a_1 \cdots a_j \star [a_k \cdots a_m] & \text{if } \textsc{FirstBackoff}(a_1 \cdots a_m) = j < m, \\ & \textsc{LMState}(a_1 \cdots a_m) = [a_k \cdots a_m] \\ a_1 \cdots a_m & \text{if } \textsc{FirstBackoff}(a_1 \cdots a_m) = m \end{cases}$$

$$\textsc{LMState}(a_1 \cdots a_m) = \begin{cases} \textsc{LMState}(a_k a_{k+1} \cdots a_m) & \text{if } k \neq 1, \\ & \mathit{type}(a_k) = [.], \forall_{i>k} \mathit{type}(a_k) \neq [.] \\ \textsc{InModel}(a_1 \cdots a_m) & \text{else} \end{cases}$$

$$\textsc{InModel}(a_i \cdots a_m) = \begin{cases} [a_i \cdots a_m] & \text{if LM contains } a_1 \cdots a_m \\ [] & \text{if } |a_i \cdots a_m| = 0 \\ \textsc{InModel}(a_{i+1} \cdots a_m) & \text{else} \end{cases}$$

$$\textsc{FirstBackoff}(a_1 \ldots a_m) = \max\{i = 0 \cdots m | \text{ LM contains } a_1 \cdots a_i\}$$

Figure 4.1: Modified q function to generate state based LM contexts to identify chart items when using a language model that is capable of returning information regarding the existence of n-grams and their shortened histories in the model. The helper functions LMState, InModel and FirstBackoff are described in the text. The *type* function checks whether a token a_k is a state, indicated by [.].

where j is the number of consecutive terminal symbols from $a_1 \cdots a_m$ that actually exist in the LM. j is determined by the FirstBackoff function. This function takes a sequence of terminals and states, returning the length of the longest terminal-only prefix that can be found in the LM. The second half of the context is the longest state found in the model when calculating p for $a_1 \cdots a_m$. In the case where all of $a_1 \cdots a_m$ is found in the model, the resulting LM context is $a_1 \cdots a_m$. The longest state found in the model is calculated by the LMState function, which seeks past the last state in $a_1 \cdots a_m$ to process the remaining terminal words with the last state as left context.

Using a state-based language model context, instead of representing $n-1$ words of history to identify language model context, allows us to use the same large, higher order language models typically deployed in state-of-the-art phrase-based systems. Recent work by Li and Khudanpur [2008] follows this same approach.

4.2.4 Decoding Parameters

In all of the experiments in this chapter we chose pruning parameters β_n such that further time spent decoding does not yield further improvements. Due to the large number of nonterminal labels generated by the SAMT rule extraction procedure, an additional parameter β_{lhs} is introduced, which limits that number of items share i, j. β_{lhs} is set to 1000 in our experiments. For the PSCFG based methods, we set the reordering limit (described in Section 2.6) $\beta_{reo} = 15$, for sentences up to a length of 20 words, and $\beta_{reo} = 12$ for longer sentences. This reduced reordering limit for long sentences was necessary to limit memory usage.

4.3 Language and Resource Conditions

We run experiments on three NIST language pair tracks under standard and alternative data conditions. These conditions are listed in Appendix A in Tables A.3, A.4, A.5.

For the Chinese-to-English and Arabic-to-English translation tasks, we define three configurations. The Full configuration represents the use of all resources as specified by the NIST large track task. These resources include parallel source and target data aligned as sentence pairs, mostly from broadcast news data sources, and monolingual English data used to estimate the n-gram language model. The second configuration, TargetLM, uses a LM trained from the target side of the parallel corpora only, in order to compare approaches under a relatively weaker n-gram LM. The third configuration, Target-LM-10%TM, is a simulation of a low data scenario, where only 10% of the bilingual training data is used with the language model from the TargetLM configuration.

These alternative data configurations allow us to explore the relative impact of PSCFG methods under varied data conditions. PSCFG methods apply reordering operations via rule with nonterminal symbols. These reordering operations are performed in context of the lexical symbols in each rule. Phrase-based system typically use simpler, reordering models even when they are based on specific phrases [Zens and Ney, 2006a]. As a result, the phrase-based system can be expected to rely more heavily on the n-gram language model more heavily to select fluent translations.

We test this hypothesis by checking relative improvements from PSCFG methods with both large and small language models. We want to determine if the relative improvements reported for PSCFG methods persist when large language models are available to influence the selection of reordered translations. The small data configu-

rations (Target-LM-10%TM), allow us to judge the impact of PSCFG methods when training data is limited. Since PSCFG methods represent their reordering operations via rules extracted from the training data, it is interesting to determine if the relative impact reduces when available training data is limited.

Translation quality is automatically evaluated by the BLEU score [Papineni et al., 2002] (case-sensitive, using length of the closest reference translation) on the following publicly available NIST *test* corpora: MT02, MT03, MT05, MT06, MT08. We used the NIST MT04 corpus as *development* data to train model parameters λ. All decoders used the MAP decision rule. For the purposes of stable comparison across multiple *test* sets, we additionally report a TstAvg score, which is the average of all *test* set scores. The test sets are described in Table A.6.

4.4 Analysis of Results

Results in Tables 4.1,4.2,4.3 demonstrate the relative impact of PSCFG methods over the phrase-based baseline evaluated at several alternative reordering window parameters for the Chinese, Arabic and Urdu to English tasks.

Chinese-to-English: We see consistent improvements by increasing the reordering limit in the phrase based system, increasing the limit from 4 to 12 yields 3.3% BLEU (absolute difference) on the composite TstAvg. Moving from the top performing phrase-based system to Hiero yields an additional 0.8%, and an additional 0.3% comes from using the SAMT grammar. Results on individual tests sets confirm this general trend as well, but we see significant variance in the improvements across test sets.

Our results with the Hiero system are consistent with those reported in Chiang [2007], where the PSCFG system uses $\beta_{reo} = 10$ and is compared to a phrase-based system with $\beta_{reo} = 7$. Results using the FULL configuration verify that PSCFG

Zh.En System	MT04 (Dev)	MT02	MT03	MT05	MT06	MT08	TstAvg
FULL							
Phrase $\beta_{reo} = 4$	37.5	38.0	38.9	36.5	32.2	26.2	**34.4**
Phrase $\beta_{reo} = 7$	40.2	40.3	41.1	38.5	34.6	27.7	**36.5**
Phrase $\beta_{reo} = 12$	41.3*	41.0	41.8	39.4	35.2	27.9	**37.0**
Hiero	41.6*	40.9	42.5	40.3	36.5	28.7	**37.8**
SAMT	41.9*	41.0	43.0	40.6	36.5	29.2	**38.1**
TARGET-LM							
Phrase $\beta_{reo} = 4$	35.9*	36.0	36.0	33.5	30.2	24.6	**32.1**
Phrase $\beta_{reo} = 7$	38.3*	38.3	38.6	35.8	31.8	25.8	**34.1**
Phrase $\beta_{reo} = 12$	39.0*	38.7	38.9	36.4	33.1	25.9	**34.6**
Hiero	38.1*	37.8	38.3	36.0	33.5	26.5	**34.4**
SAMT	39.9*	39.8	40.1	36.6	34.0	26.9	**35.5**
TARGET-LM, 10%TM							
Phrase $\beta_{reo} = 12$	36.4*	35.8	35.3	33.5	29.9	22.9	**31.5**
Hiero	36.4*	36.5	36.3	33.8	31.5	23.9	**32.4**
SAMT	36.5*	36.1	35.8	33.7	31.2	23.8	**32.1**

Table 4.1: Translation quality (% case-sensitive IBM-BLEU) for Chinese-to-English NIST-large systems. We mark Development corpus scores with * to indicate that the parameters of the corresponding decoder were MERT-tuned for this configuration and also used in the corresponding non-marked configurations.

Ar.En System	MT04 (Dev)	MT02	MT03	MT05	MT06	MT08	TstAvg
FULL							
Phrase $\beta_{reo} = 4$	51.7	64.3	54.5	57.8	45.9	44.2	**53.3**
Phrase $\beta_{reo} = 7$	51.7*	64.5	54.3	58.2	45.9	44.0	**53.4**
Phrase $\beta_{reo} = 9$	51.7	64.3	54.4	58.3	45.9	44.0	**53.4**
Hiero	52.0*	64.4	53.5	57.5	45.5	44.1	**53.0**
SAMT	52.5*	63.9	54.2	57.5	45.5	44.9	**53.2**
TARGET-LM							
Phrase $\beta_{reo} = 4$	49.3	61.3	51.4	53.0	42.6	40.2	**49.7**
Phrase $\beta_{reo} = 7$	49.6*	61.5	51.9	53.2	42.8	40.1	**49.9**
Phrase $\beta_{reo} = 9$	49.6	61.5	52.0	53.4	42.8	40.1	**50.0**
Hiero	49.1*	60.5	51.0	53.5	42.0	40.0	**49.4**
SAMT	48.3*	59.5	50.0	51.9	41.0	39.1	**48.3**
TARGET-LM, 10%TM							
Phrase $\beta_{reo} = 7$	47.7*	59.4	50.1	51.5	40.5	37.6	**47.8**
Hiero	46.7*	58.2	48.8	50.6	39.5	37.4	**46.9**
SAMT	45.9*	57.6	48.7	50.7	40.0	37.3	**46.9**

Table 4.2: Translation quality (% case-sensitive IBM-BLEU) for Arabic-to-English NIST-large systems. We mark dev. scores with * to indicate that the parameters of the corresponding decoder were MERT-tuned for this configuration and also used in the corresponding non-marked configurations.

System	UrEn-Dev	UrEn-MT08
Phrase $\beta_{reo} = 4$	12.8	**18.1**
Phrase $\beta_{reo} = 7$	14.2	**19.9**
Phrase $\beta_{reo} = 10$	14.8*	**20.2**
Phrase $\beta_{reo} = 12$	15.0	**20.1**
Hiero	16.0*	**22.1**
SAMT	16.1*	**22.6**

Table 4.3: Translation quality (% case-sensitive IBM-BLEU) for Urdu-English NIST-large. We mark Dev. scores with * to indicate that the parameters of the corresponding decoder were MERT-tuned for this configuration.

methods retain their relative improvements when using a large scale, high order, n-gram model. Furthermore, SAMT labels bring small, but consistent benefits even in the presence of strong language models.

Moving down to the TARGET-LM configuration, we still see that the size of the reordering window plays a large role in the translation quality of the phrase-based system, but comparing absolute scores against the FULL configuration shows a significant drop in translation quality (2.4% at each reordering window size on TstAvg). The Hiero system is no longer consistently better than the phrase-based system, specifically on MT02,03,04,05, where Hiero scores are lower. SAMT perform relatively better than Hiero under TARGET-LM configuration, perhaps due to the corrective influence of the syntactic labels. This confirms our intuition that the Hiero system relies more heavily on strong LMs than SAMT to navigate its decoding search space.

The TARGET-LM-10%TM configurations simulate a low data scenario. The phrase-based system degrades more than Hiero from the reduction in parallel train-

ing data — from 34.4 to 31.5 (3.1%) for phrase-based, compared to 34.4 to 32.4 (2%) for Hiero. This result is surprising, since PSCFG systems rely on rules learned from data to perform reordering operations. SAMT does not continue to outperform Hiero at this configuration. Data sparsity is the likely explanation for this relative drop in SAMT performance. Syntactic labels fragment a single rule form into several alternative labelings, resulting in poorly estimated features.

Arabic-to-English: Neither Hiero nor SAMT show consistent improvements over the phrase-based system for the Arabic-to-English translation task in Table 4.2. In contrast to the Chinese-to-English task, increasing the phrase-based reordering window beyond 4 does not yield significant improvements in any configuration. The n-gram LM still has a significant impact in Arabic-to-English, with relative drops of more that 3.4% from FULL to TARGET-LM, measured on the phrase-based system at reordering level 9. These results indicate that translational equivalence can be modeled locally for Arabic-to-English translation. The n-gram language model serves this purpose by influencing the selection of locally reordered translations. This focus on local translation effects indicates that the potential for context sensitive long distance reordering, via PSCFG methods, is limited for Arabic-to-English translation.

Urdu-to-English: Table 4.3 shows clear improvements when moving from phrase-based to PSCFG based systems. Like Chinese-to-English translation, increasing the reordering window has a significant positive effect on the BLEU score (18.1 to 20.1 from window 4 to 12). SAMT brings added benefits above Hiero but the improvements are relatively small and only measured on a single data set.

Summary over Languages: Results on Chinese-to-English and Urdu-to-English indicate that PSCFG methods outperform phrase-based methods when significant, long distance reordering effects are required to generate fluent translations. The techniques described in the previous section allow us to deploy PSCFG systems that

can take advantage of the large n-gram language models, traditionally used in phrase-based systems. There are small, but consistent, improvements in translation quality when using the SAMT grammar, which use nonterminal labels from target language parse trees. Results for Arabic-to-English translation confirms the importance of the n-gram LM in selecting locally reordered translations, but also highlights the fact that PSCFG decoders are sometimes unable to reproduce the quality of a phrase-based system.

Expressiveness of Search: Based on our comparison, we would like to know how much of the improvements we see from PSCFG methods come from performing reorderings that the phrase-based system cannot generate. In order to answer this, we attempt to regenerate translations produced by the Hiero system with the phrase-based decoder, by limiting the phrases and reordering operations applied during decoding to those that match the desired translation from the Hiero system. By forcing the phrase-based system to follow derivations consistent with the Hiero output, we can determine whether the phrase-based system could generate the same output. We used the Chinese-to-English NIST MT06 test set (1664 sentences) for this experiment. Out of the hierarchical system's translations, 1466 (88%) were generable by the phrase-based system. The relevant part of a sentence for which the Hiero translation was not phrase-based generable is shown in Figure 4.2. The reason for the failure to generate the translation is rather uninteresting. While the Hiero system is able to delete the Chinese word meaning *already* using the rule spanning [27-28], the phrase-based system has to account for this Chinese word in a phrase combining the previous word (Chinese for *epidemic*) or the following word (Chinese for *outbreak*).

Out of the generable translations, 1221 (83%) had a higher cost than the phrase-based system's preferred output. In other words, the phrase-based system does not

```
Source:      ... 怀疑 (suspect) 疫情 (epidemic) 已经 (already) 爆发 (outbreak) 的 ('s) 消息 (news) .
Reference:   ... news ... about suspicions of breakouts of the epidemic.
Phrasebased:... the epidemic has already broke the news of doubts.
Hierarchical system:
        ( [26-32: @X -> @X 的 @X . / the @X^2 of the @X^1 .]   the
            ( [31-31: @X -> 消息 / news]   news
            ) of the
            ( [26-29: @X -> 怀疑 @X / suspected @X^1]   suspected
                ( [27-29: @X -> @X 爆发 / outbreak of the @X^1]   outbreak of the
                    ( [27-28: @X -> @X 已经 / @X^1]
                        ( [27-27: @X -> 疫情 / epidemic]   epidemic
                        )
                    )
                )
            ) .
        )
```

Figure 4.2: Example from NIST MT06 for which the Hiero system's best derivation was not generable by the phrase-based system. The Hiero system's decoding derivation contains the translation in its leaves in infix order (shaded). Each non-leaf node denotes an applied PSCFG rule of the form: [Source position span, Left-hand-side >source/target]. Nonterminal markers are indicated by @ symbols with their index in γ.

prefer these forced translations due to the phrase-based *model*. These results indicate that a phrase-based system with sufficiently powerful reordering features and LM might be able to narrow the gap with the Hiero system.

4.5 Conclusions and Contributions

In this chapter we performed a systematic comparison between PSCFG systems (Hiero and SAMT) and a strong phrase-based baseline. This comparison addressed some of the major questions regarding the relative improvements from PSCFG methods.

For language pairs with significant long distance reordering (like Chinese-to-English and Urdu-to-English): we showed relative improvements from PSCFG methods when compared to a strong phrase-based model. These improvements persist even when using strong n-gram LMs, and that syntactic labels from SAMT can

bring small but consistent additional benefits. We also evaluated the space of derivations explored by the Hiero system and showed that the phrase-based system, with sufficiently long distance reordering operations, could generate the same translations with a more powerful reordering model.

For a language pair (Arabic-to-English) with limited long distance reordering: We showed that PSCFG methods bring little additional benefit over phrase-based approaches and have the potential to reduce translation quality.

The results in this chapter highlight both the potential and risk in PSCFG methods. PSCFG methods are effective at modeling long distance reordering effects, but the introduction of syntactic labels partition the search space, requiring more pruning and increased decoding time. In Chapter 5 our focus shifts to the problem of making the best use of the syntactic labels in the SAMT grammar.

CHAPTER 5

Making Better Use of PSCFG Labels: Preference Grammars

Nonterminal labels can be viewed as a non-local feature during decoding, propagating summary information about a derivation. Traditional PSCFGs view labels as a hard constraint which is enforced during decoding; nonterminal labels of rules must match at the derivation node that they share. In this chapter we propose a novel grammar formalism and associated syntactic translation model feature that represents syntactic constraints as "soft" preferences, allowing the importance of following syntactic constraints to be learned alongside other features and partially mitigating the effects of the MAP approximation. Most of the work in this chapter is to appear in Venugopal et al. [2009].

5.1 Motivation

Nonterminal labels constrain the application of rules during decoding, requiring that rules share the same nonterminal label at their shared node in a derivation. This constraint is represented explicitly in the PSCFG formalism. In Section 2.6, we stated this constraint redundantly in our definition of $p(d)$ by using the feature $p_{\text{syn}}(d)$:

$$p_{\text{syn}}(d) \quad = \quad \begin{cases} 1 & \text{if } d \text{ respects label constraints} \\ 0 & \text{otherwise} \end{cases}$$

to assign zero weight to derivations that do not obey this constraint.

Selecting the maximally weighted derivation using a grammar with nonterminal labels requires maintaining the lhs label of a derivation as an element of the chart item. As with the n-gram LM feature p_{LM}, the introduction of additional item elements results in a trade-off between maintaining non-local information versus searching through a more fragmented search space. Under the MAP approximation, different derivations that represent the same translation compete with each other. Introducing labels further aggravates the impact of this approximation. The more Bayesian approach of finding the most probable *translation* (integrating out the derivations) instantiates a NP-hard inference problem even for simple word-based models [Knight, 1999]; for grammar-based translation it is known as the consensus problem [Casacuberta and de la Higuera, 2000, Sima'an, 2002].

The SAMT approach ([Zollmann and Venugopal, 2006], Section 2.5) relies on target language parse trees to generate nonterminal labels, resorting to heuristically generated labels for non-constituent phrases. While this choice of labeling strategy results in a positive impact of translation quality for some languages (Section 4.4), the PSCFG formalism does not account for variance in parser quality or choice of labeling heuristics. The label constraint enforced by $p_{\text{syn}}(d)$ cannot be influenced on a per language or task basis to improve translation quality.

The question then is, how can we make the best use of syntactic labels?

In this chapter, we will propose a novel technique that aims to find the most probable *equivalence class* of *unlabeled* derivations, rather than a single labeled derivation, reducing the fragmentation problem. Solving this problem exactly is still a NP-hard

consensus problem, but we provide approximations that build on well-known PSCFG decoding methods. Our model falls between PSCFGs that extract nonterminal labels from parse trees and treat them as part of the derivation [Zollmann and Venugopal, 2006], and unlabeled hierarchical structures [Chiang, 2005]. We treat nonterminal labels as random variables chosen at each node, with each (unlabeled) rule expressing "preferences" for particular labels, learned from data. We use these preferences to estimate $p_{\text{syn}}(d)$ as a real-valued feature function for an unlabeled derivation, allowing the corresponding feature weight in λ to be learned via MERT [Och, 2003]. This feature will reflect the probability that an unlabeled derivation can be generated under syntactic constraints.

5.2 Related Work

Nonterminal labels and label annotations have been used extensively in the literature. Galley et al. [2004, 2006] identify PSCFG rules with syntactically motivated labels, while Huang and Knight [2006] re-label the target language parse trees in training to reflect important syntactic distinctions that are helpful to produce fluent target output. Examples of additional annotations include lexicalization of common functional nonterminal labels and adding number markers to nouns. The parsers of Collins [1999], Charniak [2000], Klein and Manning [2003] attribute their success to annotations made to the PCFGs directly licensed by manually constructed treebanks, while Petrov et al. [2006] and Matsusaki et al. [2005] propose techniques to automatically learn label annotations.

There have been significant efforts in both the parsing and machine translation literature to address the impact of the MAP approximation and the choice of labels in their respective models. We survey the work most closely related to our approach. The annotations generated by the techniques in Matsusaki et al. [2005], Petrov et al.

[2006] need to be removed in order to evaluate the quality of the output parse tree. Their annotations add an additional level of structure that must be marginalized during search. They demonstrate improvements in parse quality only when a variational approximation is used to select the most likely *unannotated* tree rather than simply stripping annotations from the MAP annotated tree.

May and Knight [2006] demonstrate an algorithm to determinize weighted finite tree automata, reducing spurious ambiguity in the grammar by summing up the weighted alternative derivations that form the same tree. This *spurious ambiguity* problem is particularly severe in methods like Data Oriented Parsing [Scha, 1990] which extract overlapping features (grammar rules) from a single tree. May and Knight [2006] show improvements in translation quality when applying their technique to determinize packed forests from a syntax-based SMT system.

Kumar and Byrne [2004] use the Minimum Bayes Risk decision rule to select the lowest risk (highest BLEU score) translation rather than derivation from a K-best list. Tromble et al. [2008] extend this work to lattice structures. These approaches only marginalize over alternative candidate derivations generated by a MAP-driven decoding process.

Zens and Ney [2006b] extend the concept of word posteriors from speech recognition [Mangu et al., 1999] to the n-gram level. If we consider the whole sentence as the n-gram, the posterior computes $\sum_h P(e, h|f)$, where h refers to alternative segmentations of the source sentence into phrases. Zens and Ney [2006b] compute n-gram posteriors on K-best lists and augments a log-linear model of translation with these additional posterior features to rescore the K-best list. Despite the relative paucity of alternative translations seen in the K-best list (compared to the full search space), results in Zens and Ney [2006b] demonstrate the value of introducing features that mitigate the impact of the maximum approximation.

Smith and Smith [2007] suggest a novel approach to introduce labels into the dependency parsing task. By introducing labels on the *edges* (rather than on nodes), they are able to marginalize over alternative labels in composing dependency rules.

More recently, work by Blunsom et al. [2007] propose a purely discriminative model whose decoding step approximates the selection of the most likely translation via beam search.

In our work, we focus on approximating the selection of the most likely unlabeled derivation during search, rather than as a post-processing operation [Kumar and Byrne, 2004, Tromble et al., 2008, May and Knight, 2006]. These post-processing methods might further improve this approximation, at some computational expense.

5.3 The Effect of Labels

We have seen evidence in Chapter 4 that syntactic labels can have a positive impact on translation quality. The primary downside to label usage (using the SAMT labeling procedure from Section 2.5), is the increase in the number of chart items created during decoding. In order to design the feature p_{syn}, we studied rules in the SAMT grammar to identify cases where a single rule form has many alternative labelings.

As a first step, we separated rules into three categories; rules without right-hand-side nonterminal symbols ("Phrases"), rules with nonterminal symbols on the left or right boundary of the source side α ("Open" rules), and rules with nonterminal symbols that are surrounded by terminal symbols in α ("Closed" rules). For each rule type, we measured the average number of unlabeled translations per source side and the average number of labeled translations per source side. The results of this analysis, using a SAMT grammar trained on the IWSLT corpus, is shown in Table 5.1.

Comparing the number of unlabeled vs labeled translations per unique α in the

Type	#Source	#Unlabeled translations	Unlabeled / #Source	Labeled translations	Labeled / #Source
Phrase	472568	963530	2.03	987894	2.09
Closed	1752575	3054183	1.74	3389282	1.93
Open	1618848	3750855	2.31	7351514	4.54

Table 5.1: Number of unique sources α, unlabeled and labeled translations and ratios of unlabeled and labeled translations for Phrase, Open and Closed rules from a grammar trained on the IWSLT corpus.

grammar shows us that the Open rules have significantly more label configurations per translation than Closed rules and Phrase rules. Phrase rules are at the opposite end of the spectrum, where they have on average 2.03 unlabeled translations per source sequence, and 2.09 labeled translations per source side. Closed rules have slightly more label variation per translation, but are still significantly less than Open rules. These results shows that the real source of alternative label variation comes from the Open rules. One potential solution to reducing the number of chart items during decoding, would be to discard Open rules. We could try to compensate for these rules by using a longer initial phrase length (β_{len}) during training and reordering limit (β_{reo}) during decoding. The intuition is that by using a larger β_{len}, we might be able to find a Closed rule that represented the same reordering operation as a discarded Open rule.

In Table 5.2, we show how translation quality using the Hiero grammar on the IWSLT task degrades when removing Open rules.

Removing open rules ("Remove Open"), results in a significant drop in translation quality, both on development and test data. Increasing to $\beta_{len}, \beta_{reo} = 15$, increases translation quality but does not come close to baseline performance, while $\beta_{len}, \beta_{reo} =$

System	Dev. BLEU	Test. BLEU
All rules $\beta_{len}, \beta_{reo} = 10$	0.280	0.370
Remove Open $\beta_{len}, \beta_{reo} = 10$	0.246	0.346
Remove Open $\beta_{len}, \beta_{reo} = 15$	0.254	0.350
Remove Open $\beta_{len}, \beta_{reo} = 20$	0.243	0.334
Remove Open $\beta_{len}, \beta_{reo} = 15$ w/Delete	0.254	0.345

Table 5.2: Comparing translation quality when removing Open rules and increasing the initial phrase length β_{len} and maximum reordering limit β_{reo} during decoding to compensate. Translation quality is measured by the BLEU score (mixed case), on the IWSLT 2006 (Dev.) and 2007 (Test) corpora.

20 reduces translation quality. We also considered re-introducing those Open rules that have unaligned source words that are deleted in translation (these rules would not be generated according to Chiang [2005]), as indicated by "w/Delete", but this did not recover the baseline score. These results show that Open rules play an important role within the decoder, concatenating the translations of long spans without resorting to the purely monotonic glue rule. Using an Open rule allows the derivation to be used with other non-glue rules. In the SAMT grammar, these Open rules have many alternative labelings, aggravating the impact of the MAP approximation. We now design the feature p_{syn} to mitigate the impact of labels on the MAP approximation, using a Open rule as a motivating example below.

5.4 Preference Grammars

We extend the PSCFG formalism to include soft "label preferences" for unlabeled rules that correspond to alternative labelings that have been encountered in training data for the unlabeled rule form. These preferences, estimated via relative frequency

counts from rule occurrence data, are used to estimate the feature $p_{\text{syn}}(d)$, the probability that an unlabeled derivation can be generated under traditional syntactic constraints.

In classic PSCFG, $p_{\text{syn}}(d)$ enforces a hard syntactic constraint (Equation 2.11). In our approach, label preferences influence the value of $p_{\text{syn}}(d)$. We follow an example below that demonstrates the impact of labels on the MAP approximation.

Motivating example: Consider the following labeled Chinese-to-English PSCFG rules:

(4) S → 我 在 哪 能 VB_1 #
a place where I can VB_1

(3) S → 我 在 哪 能 VP_1 #
a place where I can VP_1

(2) SBAR → 我 在 哪 能 VP_1 #
a place where I can VP_1

(1) FRAG → 我 在 哪 能 AUX_1 #
a place where I can AUX_1

(8) VB → 吃饭 # *eat*

(1) VP → 吃饭 # *eat*

(1) NP → 吃饭 # *eat*

(10) NN → 吃饭 # *dish*

where the numbers are frequencies of the rule from the training corpus. In classical PSCFG we can view the nonterminals labels mentioned in the rules as hard constraints on which rules can be used to expand a particular node; e.g., a VP can only be expanded by a VP rule. In Equation 2.10, $p_{\text{syn}}(d)$ explicitly enforces this hard constraint. Instead, we propose softening these constraints. In the rules below, labels

are represented as soft preferences.

$$\text{(10)} \quad \text{X} \rightarrow \text{我 在 哪 能 X}_1 \text{ \#} \quad \textit{a place where I can X}_1$$

$$\left\{ \begin{array}{rcl} p(H_0 = \text{S}, H_1 = \text{VB} \mid r) & = & 0.4 \\ p(H_0 = \text{S}, H_1 = \text{VP} \mid r) & = & 0.3 \\ p(H_0 = \text{SBAR}, H_1 = \text{VP} \mid r) & = & 0.2 \\ p(H_0 = \text{FRAG}, H_1 = \text{AUX} \mid r) & = & 0.1 \end{array} \right\}$$

$$\text{(10)} \quad \text{X} \rightarrow \text{吃饭 \# } \textit{eat}$$

$$\left\{ \begin{array}{rcl} p(H_0 = \text{VB} \mid r) & = & 0.8 \\ p(H_0 = \text{VP} \mid r) & = & 0.1 \\ p(H_0 = \text{NP} \mid r) & = & 0.1 \end{array} \right\}$$

$$\text{(10)} \quad \text{X} \rightarrow \text{吃饭 \# } \textit{dish}$$

$$\left\{ \begin{array}{rcl} p(H_0 = \text{NN} \mid r) & = & 1.0 \end{array} \right\}$$

Each unlabeled form of the rule has an associated distribution over labels for the nonterminals referenced in the rule; the labels are random variables H_i, with H_0 the left-hand-side label. These unlabeled rule forms are packed representations of the original labeled PSCFG rules. Rule features $h_i(r)$ can now be estimated as relative frequencies from the labelings of the base, unlabeled rule. Our primary contribution is how we compute $p_{\text{syn}}(d)$ for derivations built from preference grammar rules. By using $p_{\text{syn}}(d)$ as a feature in the log-linear model, we allow can use a discriminative training procedure like Minimum Error Rate Training (MERT) [Och, 2003] to evaluate the importance of syntactic structure relative to other features.

The example rules above highlight the potential for $p_{\text{syn}}(d)$ to affect the choice of translation. The translation of the Chinese word sequence 我 在 哪 能 吃饭 can be performed by expanding the nonterminal in the rule *a place where I can* X_1 with either *eat* or *dish*. The Hiero grammar would allow either expansion, relying on features

like $p_{\text{LM}}(d)$ to select the best translation since both expansions occurred the same number of times in the data.

The SAMT grammar would immediately reject the rule generating *dish* due to hard label matching constraints, but would produce three identical, competing derivations. Two of these derivations would produce S as a root symbol, while one derivation would produce SBAR. The two S-labeled derivations compete, rather than reinforce the choice of the word *eat*, which they both make. They will also compete for consideration during chart item pruning.

The rule preferences indicate that VB and VP are both valid labels for the rule translating to *eat*, and both of these labels are compatible with the arguments expected by *a place where I can* X_1. Alternatively, *dish* produces a NN label which is not compatible with the arguments of this higher-up rule. We design $p_{\text{syn}}(d)$ to reflect compatibility between two rules (one expanding a right-hand side nonterminal in the other), based on label preference distributions.

We now define the preference grammar formalism that will represents rule label preference distributions.

5.5 Preference Grammars Formalism

Probabilistic synchronous context-free *preference* grammars are defined as PSCFGs with the following additional elements:

- $\mathcal{H}$: a set of implicit labels, not to be confused with the *explicit* label set $\mathcal{N}$.
- π: $\mathcal{H} \rightarrow \mathcal{N}$, a function that associates each implicit label with a single explicit label. We can think of $\mathcal{H}$ symbols as refinements of the nonterminals in $\mathcal{N}$ [Matsusaki et al., 2005].

- For each rule r, we define a probability distribution over vectors $\vec{h}$ of implicit label bindings for its nonterminals, denoted $p_{\text{pref}}(\vec{h} \mid r)$. $\vec{h}$ includes bindings for the lhs nonterminal (h_0) as well as each right-hand side nonterminal ($h_1, ..., h_{|\vec{h}|}$). Each $h_i \in \mathcal{H}$.

In this work, we define:

- $\mathcal{N} = \{S, X\}$
- $\mathcal{H} = \{\text{NP}, \text{DT}, \text{NN} \cdots\} = \mathcal{N}_{\text{SAMT}}$

where $\mathcal{N}$ corresponds to the generic labels of Chiang [2005] and $\mathcal{H}$ corresponds to the syntactically motivated SAMT labels from Zollmann and Venugopal [2006], and π maps all elements of $\mathcal{H}$ to X. We will use hargs(r) to denote the set of all $\vec{h} = \langle h_0, h_1, ..., h_k \rangle \in \mathcal{H}^{k+1}$ that are valid bindings for the rule with nonzero preference probability. Glue rules are not assigned preferences and do not participate in the computation of the $p_{\text{syn}}(d)$ feature. Unknown words are assigned uniform preferences over the Penn Treebank noun labels NN, NNS, NP, NNP.

Grammar Formalism: The formalism presented here can be parameterized to represent several of the PSCFGs commonly applied in the literature. We list these PSCFGs and the appropriate parameters in our formalism below, using example rules to demonstrate the mapping between rules in the PSCFG and our preference grammar formalism.

Purely Hierarchical (single nonterminal) grammar [Chiang, 2005]:

- $\mathcal{N} = \{\text{S}, \text{X}\}$
- $|\mathcal{H}| = 0$
- $p_{pref}(\vec{h}|r) = \emptyset \ \forall r$

Standard SPCFGs derived from treebanks ([Galley et al., 2004, 2006]:

- $\mathcal{N} = \{\mathrm{S}, \mathrm{X}\}$
- $\mathcal{H} = \{\mathrm{NP}, \mathrm{DT}, \mathrm{NN} \cdots \in \mathcal{N}_{PennTreebank}\}$
- If $r : \mathrm{NP} \rightarrow \mathrm{DT}, \mathrm{NN}$
  - $p_{pref}(\vec{h}|r) = p_{pref}(H_0 = \mathrm{NP}, H_1 = \mathrm{DT}, H_1 = \mathrm{NN} \mid r) = 1.0$

where $\mathcal{N}_{PennTreebank}$ represent the set of nonterminals represented in the Penn Treebank [Marcus et al., 1993]. Alternatively, the grammars above can be represented using explicit labels only.

Multiple nonterminals with nonterminal specific hidden labels [Matsusaki et al., 2005, Petrov et al., 2006]:

- $\mathcal{N} = \{\mathrm{NP}, \mathrm{DT}, \mathrm{NN} \cdots \in \mathcal{N}_{PennTreebank}\}$
- $\mathcal{H} = x, y, z \cdots$ where $|\mathcal{H}|$ represents the number of latent annotations each nonterminal label can have.
- If $r : \mathrm{NP}[x] \rightarrow \mathrm{DT}[y]\mathrm{NN}[z]$
  - $p_{pref}(\vec{h}|r) = p_{pref}(H_0 = \mathrm{x}, H_1 = \mathrm{y}, H_1 = \mathrm{z} \mid r) = 1.0$

Note that while our formalism is a generalization of those described above, the choice of translation model and the independence assumptions encoded in the model determine the distribution assigned to the resulting derivations. Design decisions in the computation of $p_{\mathrm{syn}}(d)$ can be made to simulate the models discussed above. We now describe how preference distributions $p_{\mathrm{pref}}(\vec{h}|r)$ from each rule used in d are used to compute $p_{\mathrm{syn}}(d)$.

5.6 Computing Feature $p_{\text{syn}}(d)$

Let us view a derivation d as a collection of nonterminal tokens n_j, $j \in \{1, ..., |d|\}$. Each n_j takes an explicit label in $\mathcal{N}$. The score $p_{\text{syn}}(d)$ is a product, with one factor per n_j in the derivation d:

$$p_{\text{syn}}(d) = \prod_{j=1}^{|d|} \phi_j \tag{5.1}$$

Each ϕ_j factor considers the two rules that n_j participates in. We will refer to the rule above nonterminal token n_j as $\underline{r}_j$ (the nonterminal is a child in this rule) and the rule that expands nonterminal token j as $\overline{r}_j$.

The intuition is that derivations in which these two rules agree (at each j) about the implicit label for n_j, which is in $\mathcal{H}$, are preferable to derivations in which they do not. Rather than making a decision about the implicit label, we want to reward $p_{\text{syn}}(d)$ when $\underline{r}_j$ and $\overline{r}_j$ are consistent. Our way of measuring this consistency is an inner product of preference distributions:

$$\phi_j \propto \sum_{h \in \mathcal{H}} p_{\text{pref}}(h \mid \underline{r}_j) p_{\text{pref}}(h \mid \overline{r}_j) \tag{5.2}$$

This is not quite the whole story, because $p_{\text{pref}}(\cdot \mid r)$ is defined as a joint distribution of *all* the implicit labels within a rule; the implicit labels are not independent of each other. Indeed, we want the implicit labels within each rule to be mutually consistent, so that they to correspond to one of the rule's preferred labelings, for both hargs($\underline{r}$) and hargs($\overline{r}$).

Our approach to calculating $p_{\text{syn}}(d)$ within the dynamic programming algorithm is to recursively calculate preferences for each chart item based on (a) the smaller items used to construct the item and (b) the rule that permits combination of the smaller items into a larger one. We describe how the preferences for chart items are calculated. Let a chart item be denoted $[X, i, j, u, \cdots]$, where $X \in \mathcal{N}$ and i and j are

positions in the source sentence, and:

$$u : \{h \in \mathcal{H} \mid \pi(h) = X\} \rightarrow [0, 1]$$

(where $\sum_h u(h) = 1$) denotes a distribution over possible X-refinement labels. We will refer to it below as the *left-hand-side preference distribution*. Additional information, such as language model context $\tilde{e}$, may also be included in the chart item.

The simplest case is for a nonterminal token n_j that has no nonterminal children. Here the left-hand-side preference distribution is given by

$$u(h) = p_{\text{pref}}(h \mid \overline{r}_j) \ .$$

and we define the $p_{\text{syn}}(d)$ factor to be $\phi_j = 1$.

Now consider the dynamic programming step of combining an already-built item $[X, i, j, u, ...]$ rooted by explicit nonterminal X, spanning source sentence positions i to j, with left-hand-side preference distribution u, to build a larger item rooted by Y through a rule $r = Y \rightarrow \langle \gamma X_1 \gamma', \alpha X_1 \alpha', w \rangle$ with preferences $p_{\text{pref}}(\cdot \mid r)$.[1] The new item will have signature $[Y, i - |\gamma|, j + |\gamma'|, v, ...]$. The left-hand-side preferences v for the new item are calculated as follows:

$$\begin{aligned} v(h) &= \frac{\tilde{v}(h)}{\sum_{h'} \tilde{v}(h')} \quad \text{where} \\ \tilde{v}(h) &= \sum_{h' \in \mathcal{H} : \langle h, h' \rangle \in \text{hargs}(r)} p_{\text{pref}}(\langle h, h' \rangle \mid r) \times u(h') \end{aligned} \tag{5.3}$$

Renormalizing keeps the preference vectors on the same scale as those in the rules. The $p_{\text{syn}}(d)$ factor ϕ, which is factored into the value of the new item, is calculated as:

$$\phi = \sum_{h' \in \mathcal{H} : \langle h, h' \rangle \in \text{hargs}(r)} u(h') \tag{5.4}$$

[1] We assume for the discussion that $\alpha, \alpha' \in \mathcal{T}_S^*$ and $\gamma, \gamma' \in \mathcal{T}_T^*$. If there are multiple nonterminals on the right-hand side of the rule, we sum over the longer sequences in hargs(r) and include appropriate values from the additional "child" items' preference vectors in the product.

$$
\begin{aligned}
\tilde{v}(\mathrm{S}) &= p_{pref}(\langle h = \mathrm{S}, h' = \mathrm{VB}\rangle \mid \underline{r})u(\mathrm{VB}) + p_{pref}(\langle h = \mathrm{S}, h' = \mathrm{VP}\rangle \mid \underline{r})u(\mathrm{VP}) \\
&= (0.4 \times 0.8) + (0.3 \times 0.1) = 0.35 \\
\tilde{v}(\mathrm{SBAR}) &= p(\langle h = \mathrm{SBAR}, h' = \mathrm{VP}\rangle \mid \underline{r})u(\mathrm{VP}) \\
&= (0.2 \times 0.1) = 0.02 \\
v &= \langle v(\mathrm{S}) = 0.35/(\tilde{v}(\mathrm{S}) + \tilde{v}(\mathrm{SBAR})), v(\mathrm{SBAR}) = 0.02/\tilde{v}(\mathrm{S}) + \tilde{v}(\mathrm{SBAR})\rangle \\
&= \langle v(\mathrm{S}) = 0.35/0.37, v(\mathrm{SBAR}) = 0.02/0.37\rangle \\
\phi_2 &= u(\mathrm{VB}) + u(\mathrm{VP}) = 0.8 + 0.1 = 0.9
\end{aligned}
$$

Figure 5.1: Calculating v and ϕ_2 for the running example in the chapter.

so that the value considered for the new item is $w \times \phi \times ...$, where factors relating to $p_{\mathrm{LM}}(d)$, for example, may also be included. Coming back to our example, if we let $\overline{r}$ be the leaf rule producing *eat* at shared nonterminal n_1, we generate an item with:

$$
\begin{aligned}
u &= \langle u(\mathrm{VB}) = 0.8, u(\mathrm{VP}) = 0.1, u(\mathrm{NP}) = 0.1\rangle \\
\phi_1 &= 1
\end{aligned}
$$

Combining this item with $\mathrm{X} \rightarrow \langle$ 我 在 哪 能 X_1 # *a place where I can* X_1 $\rangle$ as $\underline{r}_2$ at nonterminal n_2 generates a new target item with translation *a place where I can eat*, $\phi_2 = 0.9$ and v as calculated in Fig. 5.1. In contrast, $\phi_2 = 0$ for the derivation where $\overline{r}$ is the leaf rule that produces *dish*.

This calculation can be seen as a kind of single pass, bottom-up message passing inference method embedded within the usual dynamic programming search.

5.7 Decoding with Preference Grammars

As defined above, accurately computing $p_{\mathrm{syn}}(d)$ requires extending the chart item structure with the item's left-hand-side preference distribution v. For models that

$$\frac{r : X \rightarrow \langle f_{i+1}^{j}, \alpha, p_{pref}(\bar{h}|r) \rangle : w}{[X, i, j; q(\alpha)], \quad \underbrace{p_{pref}(\vec{h}|r)}_{\text{lhs preference distribution}} \quad : wp(\alpha)}$$

$$\frac{r : Z \rightarrow \langle f_{i+1}^{i_1} X_1 f_{j_1+1}^{i_2} Y_2 f_{j_2+1}^{j}, \alpha, p_{pref}(\bar{h}|r) \rangle : w \ , \ [X, i_1, j_1, \tilde{e}_1], u_1 : w_1 \ , \ [Y, i_2, j_2, \tilde{e}_2], u_2 : w_2}{[Z, i, j, q(\alpha')], v : ww_1w_2p(\alpha')\phi \quad (\text{where } \alpha' = \alpha\,[\tilde{e}_1/X_1, \ldots, \tilde{e}_2/X_2])}$$

Figure 5.2: CYK parsing with integrated n-gram LM and integrated computation of the composition preference feature $p_{\text{syn}}(d)$. The computation of ϕ and v are given in the text.

use the n-gram LM feature, the item structure would be:

$$[X, i, j, \tilde{e}, v] : w \tag{5.5}$$

Since v effectively summarizes the choice of rules in a derivation, this extension would partition the search space *further*. To prevent this partitioning, we follow the approach of Venugopal et al. [2007] (Chapter 3). Rather than including v in the item structure, we treat v as information associated with the chart item:

$$[X, i, j, \tilde{e}], v : w \tag{5.6}$$

Using this item structure, extensions to this chart item would use v from the highest weighted derivation that shares the $X, i, j, q(\alpha)$. Our intuition is that derivations that share the same LM context $\tilde{e}$ can be expected to have relatively similar syntactic behavior and thus relatively similar left-hand-side preference distributions v.

Decoding with preference grammars under this approximation is shown in Figure 5.2. In the first inference, we show the formation of a chart item from a rule that has no right-hand-side nonterminal symbols. In the second inference, two chart items with their associated distributions u are combined with a rule to form a new chart item. When rules have k nonterminal symbols, the computation of $\tilde{v}(h)$ is extended as:

$$\tilde{v}(h) = \sum_{h'_1 \cdots h'_k \in \mathcal{H}^k : \langle h, h'_1 \cdots h'_k \rangle \in \text{hargs}(r)} p_{\text{pref}}(\langle h, h'_1 \cdots h'_k \rangle \mid r) \times \prod_{i=1}^{k} u_i(h'_i) \tag{5.7}$$

where u_i is the preference distribution for the i'th antecedent item.

After first pass decoding, we follow the Top-Down approach described in Section 3.4. We perform a Tow-Down hypergraph search where the feature $p_{\text{syn}}(d)$ is recomputed while extracting a K-best list. During Top-Down search, each hyperedge is associated with a rule. When we consider alternative derivations at the tail nodes of the hyperedge, we recompute the $p_{\text{syn}}(d)$ feature based on the rule and the alternative tail items. It is therefore possible to find a higher weighted derivation due to more compatible preferences in the second pass.

Preference Pruning Parameters: As noted in Section 5.3, Open rules have a large number of alternative label configurations when extracted by the SAMT labeling procedure. As the size of the training data grows, the number of alternative label configurations grows as well. This is especially true for Open rules that are essentially agnostic about their label choices. Dealing with the number of label preferences at each rule presents serious memory and runtime challenges. Consider the sentence specific preference grammar extracted for the Chinese-to-English sentence in Table 5.3.

The grammar for this sentence from system NIST-M (defined below) includes 4654 rules but the number of label configurations for each rule varies significantly. Table 5.4 shows the rules that have the most number of label configurations in this grammar.

The Chinese-to-English rule X $\rightarrow$ $\langle$ X_1 的 X_2 # X_1 *'s* X_2 $\rangle$ has over 24K elements in hargs(r) when learned for the NIST-M task (Table 5.4). In order to limit the memory requirements of maintaining large preference distributions and the compu-

越来越 多 的 迹象 显示 , 森尼派 回教徒 可能 抵制 一 月 三十日 的 大选 , 也 可能 因为 散播 叛乱 而 被 伊拉克 有关 当局 禁 止 参加 大选 。
there are growing indications that sunni arabs will either boycott the january 30th election or will be prevented from taking part in it by the iraqi authority due to a spreading insurgency .
there has been growing evidence that sunni muslims may boycott the elections on january 30 or may be prohibited from participating in the elections by the relevant iraqi authorities due to their spreading of insurgency .
more and more evidence has shown that sunni muslims will either boycott the january 31 election or will be prevented from participating in the election by the iraqi authorities for spreading insurgency .
there are growing indications that sunni muslims may either boycott the january 30 elections or be banned from the elections by relevant iraqi authorities for spreading insurgency .

Table 5.3: An example sentence from the NIST MT06 Chinese-to-English translation task with 4 reference translations.

tational impact of iterating through these distributions to compute $p_{\text{syn}}(d)$, we limit the number of elements in $p_{pref}(\vec{h}|r)$ and number of elements in the item preference vector v via the following pruning parameters:

- β_R: This parameter limits the size of hargs(r) to the β_R top-scoring preferences, defaulting other values to zero.
- β_L: This parameter is the same as β_R but applied only to rules with no nonterminals. The stricter of β_L and β_R is applied if both thresholds apply.
- β_P: This parameter limits the number labels in *item* preference vectors (Equation 5.3) to the β_P most likely labels during decoding, defaulting other prefer-

#Label Conf.	Source γ	Target α	$max\ p_{pref}(\vec{h}\|X, \gamma, \alpha)$
24597	X_1 的 X_2	X_1 *'s* X_2	NP+JJ → NNP JJ =0.02
22879	X_1 的 X_2	X_1 X_2	JJ+NN → JJ NN=0.02
6803	X_1 的 X_2	X_1 *of* X_{-2}	NP → NP NP=0.03
6155	X_1 。	X_1 .	VP+. → VP =0.15
6043	X_1 也 X	X_1 *also* X_2	NP+RB+VBD → NP VBD =0.01
5777	X_1 而 X	X_1 X_2	CC+NP → CC NP=0.01
5082	X_1 有关 X	X_1 X_2	JJ+NNS → JJ NNS=0.01
4465	X_1 。 X	X_1 . X_2	VP+.+NNP → VP NNP =0.07
4438	X_1 多 X	X_1 X_2	CD+NNS → CD NNS=0.04
3749	X_1 的 X	*the* X_1 X_2	NP → JJ NN=0.06

Table 5.4: Example preference grammar rules from a medium size Chinese-to-English translation task, highlighting rules with the most label configuration preferences and the most likely preferences for each rule.

ences to zero.

5.8 Empirical Results

In this Section we empirically evaluate our preference grammar translation model on the IWSLT and NIST-M Chinese-to-English translation tasks as described in the Appendix A, Table A.1). For the IWSLT task, development data from the IWSLT 2003-2005 are added to the training data, IWSLT 2006 is the development data, IWSLT 2007 and 2008 are the test corpora. For the NIST-M task, MT05 is used as development data, MT06 is the test data. A 5-gram LM is built for the IWSLT task from the target side of the training data and a 4-gram LM is built for the NIST-M task based on the monolingual resources described in Appendix A.

We compare the performance of our preference grammar approach, evaluated by the BLEU score, against two baseline approaches; SAMT and Hiero. Development corpora are used to train model parameters λ via MERT. We use a variant of MERT that prefers sparse solutions where $\lambda_i = 0$ for as many features as possible. At each MERT iteration, a subset of features λ are assigned 0 weight and optimization is repeated. If the resulting BLEU score is not lower, these features are left at zero. All systems use a trigram LM during search and the full-order LM during a second Top-Down hypergraph rescoring pass. The initial phrase length and reordering limit are $\beta_{len}, \beta_{reo} = 10$. For the NIST-M task, rare rules are discarded based on their frequency in the training data. Purely lexical rules (that include no terminal symbols) that occur less than two times, or non-lexical rules that occur less than four times are discarded.

IWSLT task: We evaluate the preference grammar system "Pref." with parameters $\beta_R = 100$, $\beta_L = 5$, $\beta_P = 2$. Results comparing systems Pref. to Hiero

System	Dev BLEU (lpen) ↑	2007 BLEU (lpen) ↑	2008 BLEU (lpen) ↑	2008 WER ↓	2008 PER ↓	2008 MET. ↑	2008 GTM ↑
Hiero	28.0 (0.89)	37.0 (0.89)	45.9 (0.91)	44.5	39.9	61.8	70.7
SAMT	30.9 (0.96)	35.5 (0.94)	45.3 (0.95)	45.7	40.4	62.1	71.5
Pref.	28.3 (0.88)	38.2 (0.90)	46.3 (0.91)	43.8	40.0	61.7	71.2

Table 5.5: Translation quality scores on the IWSLT translation task, with IWSLT 2006 as the development corpora, and IWSLT 2007 and 2008 as test corpora. Each score is annotated with an ↑ if increases in the score's value correspond to increases in translation quality and a ↓ if the opposite is true. We also list length penalties for the BLEU score to show that improvements are not due to length optimizations alone.

and SAMT are shown in Table 5.5. Automatic evaluation results using the preference grammar translation model are positive. The preference grammar system shows improvements over both the Hiero and SAMT based systems on both unseen evaluation sets IWSLT 2007 and 2008. The improvements are clearest on the BLEU score (matching the MERT training criteria). On 2007 test data, Pref. shows a 1.2% (absolute) improvement over Hiero, while on the 2008 data, there is a 0.6% improvement. For the IWSLT task, we report additional automatic evaluation measures that generally rank the Pref. system higher than Hiero and SAMT. As a further confirmation, our feature selection based MERT chooses to retain λ_m (the corresponding weight for the $p_{\text{syn}}(d)$ feature) in the model. While the IWSLT results are promising, we perform a more complete evaluation on the NIST-M translation task.

NIST-M task: This task generates much larger rule preference vectors than the IWSLT task due to the size of the training corpora. We build systems with both $\beta_R = 100, 10$ varying β_P. Varying β_P isolates the relative impact of propagating alternative nonterminal labels within the preference grammar model. $\beta_L = 5$ for all NIST-M systems. Parameters λ are trained via MERT on the $\beta_R = 100$, $\beta_L = 5$, $\beta_P = 2$ system. BLEU scores for each preference grammar and baseline system are shown in Table 5.6, along with translation times on the test corpus. We also report length penalties to show that improvements are not simply due to better tuning of output length.

The preference grammar systems outperform the Hiero baseline by 0.5% on development data, and up to 0.8% on unseen test data. While systems with $\beta_R = 100$ take 15 times longer to translate the test data than Hiero, setting $\beta_R = 10$ takes 50% longer than the the SAMT based system but produces slightly better results (0.3%).

The improvements in translation quality with the preference grammar are positive, but how much of this improvement can be attributed to MERT finding a better local

System	Dev. BLEU (lpen)	Test. BLEU (lpen)	Test time (h:mm)
Baseline Systems			
Hiero	34.1 (0.99)	31.8 (0.95)	0:12
SAMT	34.7 (0.99)	32.3 (0.95)	0:45
Hiero($\lambda^\star$)	-	32.1 (0.95)	0:12
Preference Grammar: $\beta_R = 100$			
$\beta_P = 1$	-	32.5 (0.96)	3:00
$\beta_P = 2$	34.6 (0.99)	32.6 (0.95)	3:00
$\beta_P = 5$	-	32.5 (0.95)	3:20
Preference Grammar: $\beta_R = 10$			
$\beta_P = 1$	-	32.5 (0.95)	1:03
$\beta_P = 2$	-	32.6 (0.95)	1:10
$\beta_P = 5$	-	32.5 (0.95)	1:10

Table 5.6: Translation quality and test set translation time (using 50 machines with 2 tasks per machine) measured by the BLEU score for the NIST-M task. NIST 2006 is used as the development (Dev.) corpus and NIST 2007 is used as the unseen evaluation corpus (Test). Dev. scores are reported for systems that have been separately MERT trained. Pref. systems share parameters from a single MERT training. Systems are described in the text.

optimum for parameters λ? To answer this question, we use parameters $\lambda^\star$ optimized by MERT for the preference grammar system to run a purely hierarchical system, denoted Hiero($\lambda^\star$), which ignores the value of λ_m during decoding. While almost half of the improvement comes from better parameters learned via MERT for the preference grammar systems, 0.5% can be still be attributed purely to the feature $p_{\text{syn}}(d)$. In addition, MERT does not set parameter λ_m to zero, corroborating the value of the $p_{\text{syn}}(d)$ feature again. Note that Hiero($\lambda^\star$) achieves better scores than the Hiero system which was trained via MERT without $p_{\text{syn}}(d)$. This highlights the local nature of MERT parameter search, but also points to the possibility that training with the feature $p_{\text{syn}}(d)$ produced a more diverse derivation space, resulting in better parameters λ. We see a very small improvement (0.1%) by allowing the runtime propagation of more than 1 nonterminal label in the left-hand side posterior distribution, but the improvement doesn't extend to $\beta_P = 5$. Improved integration of the feature $p_{syn}(d)$ into decoding might help to widen this gap.

5.9 Analysis of a Translation

We present an example sentence from the NIST MT06 test corpus and compare the translation output from two systems. We compared output from the preference grammar system $\beta_R = 10$, $\beta_L = 5$, $\beta_P = 2$ above against output from the Hiero($\lambda^\star$) system. These systems differ *only* in the use of the preference grammar feature. All other weights in λ are identical.

We chose our sentence pair based on simple inspection of the translation ouput. Our goal was to find a relative short sentence (long enough to exhibit reordering but not too long to analyze), where translation output differed significantly across the two systems. Table 5.7 shows the source and four reference reference translations for sentence that we chose for comparison of systems.

埃及 官员 正在 调查 南部 一 座 养鸡场 突然 发生 的 鸡只 集体 暴 毙 事件 , 担心 病毒 可能 会 持续 传播 。
an egyptian official is currently investigating a southern chicken farm that saw groupings of sudden deaths ; they worry that the virus will likely continue to spread .
egyptian officials are investigating the sudden mass death of chickens on a chicken farm in the south , and there is concern that the virus might continue to spread .
egyptian officials are investigating a sudden incidence of mass chicken deaths on a chicken farm in the south , fearing that the virus may continue to spread .
egyptian officials are also investigating the sudden unexplained death of a batch of chickens in a poultry farm , raising concerns that the virus may continue to spread .

Table 5.7: Source and four reference translation for an example sentence from NIST MT06. We use this example sentence to compare translation output between the Pref. system $\beta_R = 10$, $\beta_L = 5$, $\beta_P = 2$ and the Hiero($\lambda^\star$) system.

The translation output from preference grammar system is:

a chicken farm in southern egyptian officials are investigating cases of collective 暴毙 *chickens suddenly occurred , worried that the virus may continue to spread .*

while the output from the hierarchical system Hiero($\lambda^\star$) is:

a chicken farm in southern egyptian officials are investigating the chickens of the collective 暴毙 *incident occurred suddenly and may be worried that virus continued to spread .*

While the lexical content of both system outputs are very similar, the reordering effects and therefore both the syntax and semantics of the sentence are ultimately very different. The first few words of both outputs are the same (*a chicken farm in southern egyptian officials are investigating*); the differences are in the remainder of the sentence. Overall, the preference grammar system captures the meaning of the source sentence better, indicating that the *officials* are *investigating cases of*

System	1-gram	2-gram	3-gram	4-gram	Reference
Pref.	20 / 26	13 / 25	10 / 24	7 / 23	28
Hiero($\lambda^\star$)	20 / 28	9 / 27	6 / 26	3 / 25	28

Table 5.8: N-gram precision ratios for the translation output from the Pref. system and Hiero($\lambda^\star$) .

something rather than *investigating chickens*, and clearly stating that the officials are *worried that the virus may continue to spread* ., rather than the fact that they *may be worried*. The respective BLEU precision ratios at each n-gram level are shown in Table 5.8 and confirm the relative performance of the preference grammar system.

We now analyze the rules used in the preference system's translation to highlight cases where the $p_{\text{syn}}(d)$ had an impact on the decisions made during translation. The output from the preference grammar is composed of the translations of four spans that have concatenated monotonically with the glue rule to form the complete sentence translation. For each span, we show the composition of rules that form the span's translation in Figures 5.3,5.4, 5.6. Each rule is annotated with the lhs distribution of the corresponding chart item formed by the rule.

The span translation from Figure 5.4 is illustrative of the impact of the $p_{\text{syn}}(d)$ feature. We list the rules used in Figure 5.4 with their label configurations in Table 5.5. We list the highest probability preferences until the preference(s) that were used in the example translation, which is marked by a $\star$.

Both systems, Hiero($\lambda^\star$) and Pref. use the rule X $\rightarrow \langle$ 担心 X_1 # *worried that* X_1 $\rangle$, but compose it with different derivations. In Pref. , this rule is composed with the chart item generated by the rule X $\rightarrow \langle$ X_1 可能会 #*the* X_1 *may*$\rangle$. In Figure 5.4, we can see that the resulting lhs distribution of *the virus may* is NP+MD. This lhs combines with *worried that* X_1 with non-zero probability. The Hiero($\lambda^\star$) system

combines X → ⟨ 担心 X_1# *worried that* X_1 ⟩ with X → ⟨病毒 # *virus* ⟩ directly. This combination would result in a relative larger penalty in the preference grammar system, since the corresponding label configuration in the *worried that* X_1 rule is VP/VP→ NN=0.00.

5.10 Conclusions and Contributions

In this chapter, we developed a novel grammar formalism that generalizes existing work to integrate syntactic labels into natural language tasks. Our formalism transforms hard syntactic constraints into soft preferences that can be efficiently estimated during the extraction of SAMT rules from parallel data.

These preferences are used to compute a machine translation feature ($p_{\text{syn}}(d)$) that scores unlabeled derivations, taking into account traditional syntactic constraints. Representing syntactic constraints as a feature allows MERT to train the corresponding weight for this feature relative to others in the model, allowing systems to learn the relative importance of labels for particular resource and language scenarios as well as for alternative approaches to labeling PSCFG rules. Our computation for $p_{\text{syn}}(d)$ is designed to avoid penalizing rules that are used for generic operations like concatenation and as a result, have a large number of label preferences.

We showed moderate improvements in translation quality as measured by automatic evaluation measures for both small and medium resource translation tasks. On the medium resource translation task we improve 0.3% over the SAMT baseline and 0.5% over the best performing Hiero baseline.

By moving syntactic labels into soft preference structures, we introduce an avenue by which label preferences can be automatically learned via discriminative training. Future work will focus on methods to improve on the integration of the $p_{\text{syn}}(d)$ feature

during decoding.

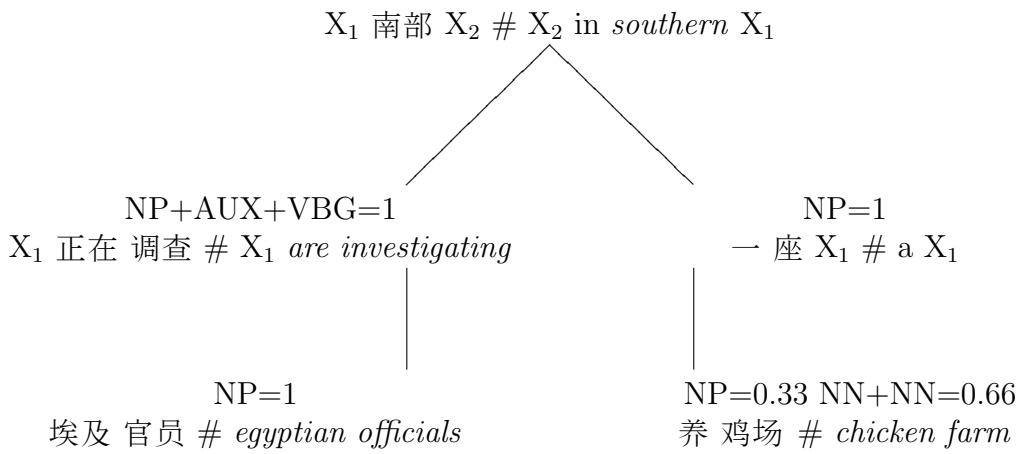


Figure 5.3: Rules applied in the preference grammar system for the example sentence in Table 5.7. The target translation generated by these rules is: *a chicken farm in southern egyptian officials are investigating*

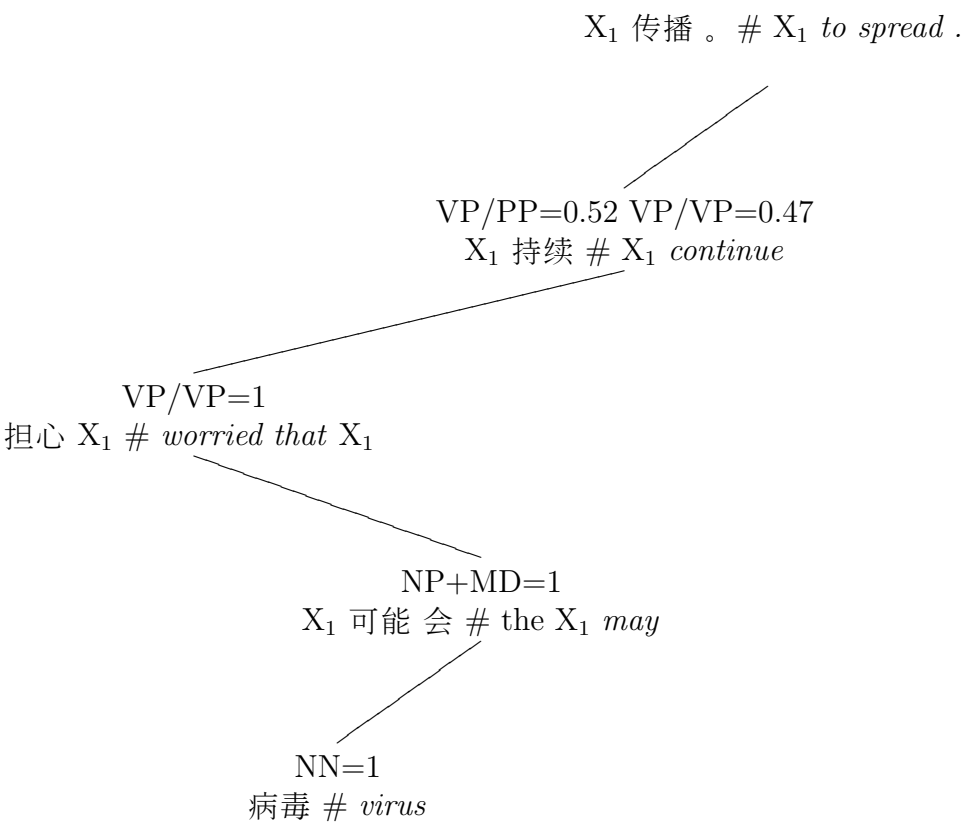


Figure 5.4: Rules applied in the preference grammar system for the example sentence in Table 5.7. The target translation generated by these rules is: *worried that the virus may continue to spread .*

$$\mathrm{X} \rightarrow \mathrm{X}_1\text{传播 } \#\mathrm{X}_1 \textit{ to spread}$$

$$\left\{ \begin{array}{rcl} p(H_0 = \mathrm{RB{+}TO{+}VB}, H_1 = \mathrm{RB} \mid r) & = & 0.22 \\ p(H_0 = \mathrm{NP{+}CC{+}VP}, H_1 = \mathrm{X} \mid r) & = & 0.11 \\ p(H_0 = \mathrm{S}, H_1 = \mathrm{NNS} \mid r) & = & 0.11 \\ p(H_0 = \mathrm{NN{+}CC{+}VP}, H_1 = \mathrm{X} \mid r) & = & 0.11 \\ \star p(H_0 = \mathrm{VP}, H_1 = \mathrm{VP/VP} \mid r) & = & 0.11 \end{array} \right\}$$

$$\mathrm{X} \rightarrow \mathrm{X}_1\text{持续 \# } \mathrm{X}_1 \textit{ continue}$$

$$\left\{ \begin{array}{rcl} p(H_0 = \mathrm{MD{+}VB}, H_1 = \mathrm{MD} \mid r) & = & 0.29 \\ p(H_0 = \mathrm{NN{+}MD{+}VB}, H_1 = \mathrm{NN{+}MD} \mid r) & = & 0.09 \\ p(H_0 = \mathrm{X}, H_1 = \mathrm{X} \mid r) & = & 0.05 \\ p(H_0 = \mathrm{NP{+}MD{+}VB}, H_1 = \mathrm{NP{+}MD} \mid r) & = & 0.03 \\ p(H_0 = \mathrm{NP{+}VBP}, H_1 = \mathrm{NP} \mid r) & = & 0.03 \\ p(H_0 = \mathrm{NNS{+}VBP}, H_1 = \mathrm{NNS} \mid r) & = & 0.03 \\ \star p(H_0 = \mathrm{VP/PP}, H_1 = \mathrm{VP/VP} \mid r) & = & 0.02 \\ \star p(H_0 = \mathrm{VP/VP}, H_1 = \mathrm{VP/VP} \mid r) & = & 0.01 \end{array} \right\}$$

$$\mathrm{X} \rightarrow \text{担心}\mathrm{X}_1 \text{ \# } \textit{worried that } \mathrm{X}_1$$

$$\left\{ \begin{array}{rcl} p(H_0 = \mathrm{VP/VP}, H_1 = \mathrm{NP} \mid r) & = & 0.09 \\ p(H_0 = \mathrm{X}, H_1 = \mathrm{X} \mid r) & = & 0.08 \\ p(H_0 = \mathrm{VP/VP}, H_1 = \mathrm{NP{+}MD} \mid r) & = & 0.04 \end{array} \right\}$$

$$\mathrm{X} \rightarrow \mathrm{X}_1\text{可能 会 \# the } X_1 \textit{ may}$$

$$\left\{ \begin{array}{rcl} \star p(H_0 = \mathrm{NP{+}MD}, H_1 = \mathrm{NN} \mid r) & = & 0.84 \end{array} \right\}$$

Figure 5.5: Rules with label configurations for rules that generated: *worried that the virus may continue to spread .* in our example sentence. For each rule we list the highest probability label preferences until the preference(s) that were used in the example translation, which is marked by $\star$.

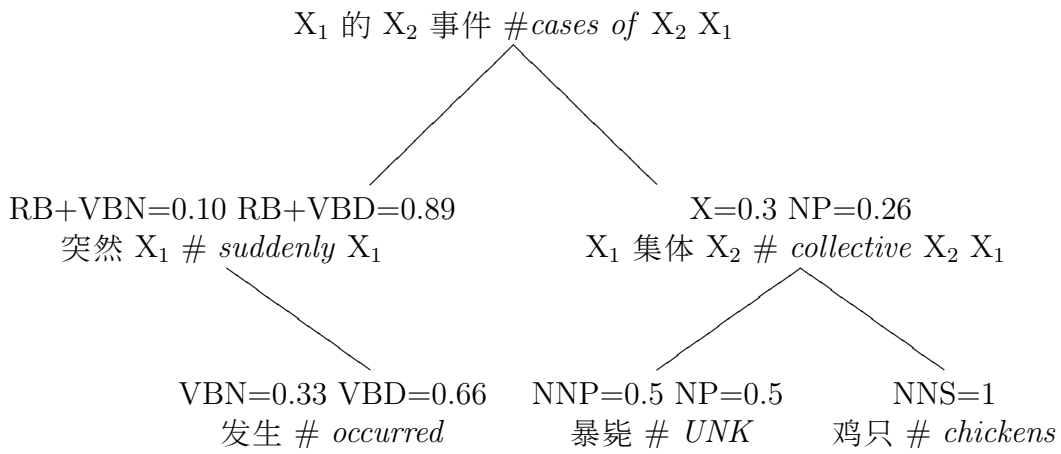


Figure 5.6: Rules applied in the preference grammar system for the example sentence in Table 5.7. The target translation generated by these rules is: *cases of collective UNK chickens suddenly occurred*.

CHAPTER 6

Conclusions and Future Work

The use of non-local features in translation models is an important step towards automatically generating human quality translations. In this work we have explored the integration and interaction between two non-local features within the PSCFG framework; the n-gram language model and nonterminal labels for PSCFG rules. We have made the following contributions in this work:

- We developed two-pass decoding algorithms to efficiently integrate n-gram LMs into PSCFG decoders (Chapter 3). Our two algorithms explore techniques to find alternative derivations from the hypergraph that results from first pass decoding. Our two-pass Top-Down algorithm (which uses Cube Pruning in its first pass) achieves the lowest model cost in our experiments 53% faster than the single pass Cube Pruning algorithm for the SAMT grammar. We use the two-pass decoding algorithms developed in Chapter 5 to integrate a non-local syntactic feature into decoding.

- In Chapter 4, we answered important questions about the impact of PSCFG methods, specifically studying the impact of nonterminal labels and large n-gram LMs on translation quality. Our large scale study showed that for language pairs with longer distance reordering effects (Chinese, Urdu to English) the Hiero grammar outperforms a strong phrase-based baseline approach with comparable reordering ability and strong n-gram LMs to inform reordering decisions.

The SAMT grammar delivers small, but consistent, additional improvements. We also show that phrase-based systems can generate 88% of the translations from the Hiero system, which suggests that improved distortion models might close the performance gap.

- In Chapter 5 we proposed a novel grammar formalism and associated feature to take advantage of nonterminal labels without aggravating the impact of the common MAP approximation that is often used in machine translation. Our approach transforms hard syntactic constraints into soft preferences that are used to estimate a syntactic consistency feature, whose weight can be discriminatively learned to improve translation quality. We demonstrate small improvements over the SAMT (0.3% BLEU) and Hiero baselines (0.5% BLEU) for a medium sized Chinese-to-English translation task and find that feature selection retains the syntactic feature in the translation model.

We hope to consider the following future research directions:

- Better search: While in this work we considered approximate first pass search, we might also consider developing admissible search heuristics that would allow more chart items to be compared and pruned during search. Initial directions would include using n-gram LMs that could report the maximum probability of a word sequence $w_1 \cdots w_n$ over all observed history contexts h i.e $\max p(w_1 \cdots_n |h)\ \forall h$. Using this probability to score the left context in $\tilde{e}$, we would have an admissible heuristic to safely compare items that differ in their LM context component but share the same span.

- Hybrid systems: Results in Chapter 4 indicate that phrase-based systems are capable of producing many (88% in our Chinese-to-English experiments) of the

translations generated by the Hiero PSCFG system but current reordering models are too weak to select these translations during search. We hope to explore techniques to apply the highly lexicalized PSCFG rules to re-rank derivations in the hypergraph produced by phrase-based search. This could result in an overall reduction in decoding time since the first pass would use phrase-based decoding.

- Better integration of the preference grammar feature p_{syn} into search: The approach used in this work makes greedy approximations to integrate p_{syn} into search. These approximations limit the impact that this feature can have in selecting derivations. We hope to explore techniques that allow multiple derivations to share the same lhs preference distribution.

- Exploring independence assumptions: In this work we try to stay close to the way traditional PSCFGs use labels to propagate non-local information about a derivation. Alternatively, we might choose to make stronger independence assumptions in the estimation of rule preferences that facilitate more efficient decoding with the p_{syn} feature.

- Discriminative training of preferences: By representing labels via soft preferences, we have created the potential to learn label preferences discriminatively like Matsusaki et al. [2005], Petrov et al. [2006] do for monolingual parsing.

Ultimately we will continue to focus on finding efficient ways to improve translation quality by using non-local features within the PSCFG framework.

APPENDIX A

Resources Used

This Appendix describes the corpora and resources used for the systems built in this thesis. Training corpora for systems identified by IWSLT and NIST-M are described in Table A.1. The IWSLT task is a limited domain dialogue travel expressions task that uses the Basic Travel Expression Corpora for training data.

The language model for the IWSLT task is built from the target side of the parallel corpus. The NIST-M task is a (M)edium resource task in the broadcast news domain. The NIST-M parallel corpus has been compiled from a variety of data sources made publicly available by the Linguistic Data Consortium for the NIST MT Evaluation competition. The NIST-M corpus includes all available data for the NIST evaluation except the UN and HK Law corpora. Sentences with less than 70 source words are selected from the remaining corpora, and the FBIS corpus is further restricted to sentences up to 60 source words. The language model for the NIST-M system is built on the target side of the selected parallel data and includes the Xinhua news stories from the years 1995 - 2006. NIST-M resources are comparable in size to medium data configurations available in several languages [Koehn, 2005]. Development and test corpora for the IWSLT and NIST-M task are described in Table A.2.

Resource conditions for experiments in Chapter 4 are shown in Tables A.3,A.4,A.5. For the Chinese and Arabic tasks, multiple data configurations are presented to simulate weaker language and translation models. We report corpus statistics and the total number (non-singleton) n-grams in the LM training data for each configuration.

For the Chinese and Arabic tasks, the language model data comes from three sources; the target side of the parallel data (448M $1\cdots5$-grams), the LDC Gigaword corpus (3.7B tokens, 2.9B $1\cdots5$-grams) and the Web 1T 5-Gram Corpus (1T tokens, 3.8B $1\cdots5$-grams).

System	Sentence Pairs	Words in Target Text
IWSLT	163K	1.2M
NIST-M	3.7M	67M

Table A.1: Training data for the IWSLT and NIST-M tasks. Target text is tokenized to separate punctuation from attached words. Chinese source text is segmented using the LDC segmenter.

Data Set	# Sentences	# Source Words	# References
IWSLT DevSet4	489	5214	16
IWSLT03	506	3404	16
IWSLT04	500	3505	16
IWSLT05	506	3826	16
IWSLT06	500	5550	7
IWSLT07	489	3166	6
IWSLT08	1014	5678	6
MT05	1082	30306	4
MT06	1664	38939	4

Table A.2: Development and test corpora for the IWSLT and NIST-M tasks.

Zh.En. Conf.	# Sentences	# Src, Tgt words	# 1..5 Grams (mono.)
FULL	15.4M	295M(zh) 336M(en)	448M + 2.9B + 3.8B
TARGET-LM	15.4M	295M(zh) 336M(en)	448M
TARGET-LM-10%TM	1.54M	29.5M(zh) 33.6M(en)	448M

Table A.3: Chinese-to-English Data configurations under which we compare phrase-based methods to PSCFG methods.

Ar.En. Conf.	# Sentences	# Src, Tgt words	# $1 \cdots 5$ Grams (mono.)
FULL	9.1M	223M(ur) 236M(en)	448M + 2.9B + 3.8B
TARGET-LM	9.1M	223M(ur) 236M(en)	448M
TARGET-LM-10%TM	0.91M	22.3M(ur) 23.6M(en)	448M

Table A.4: Arabic-to-English Data configurations under which we compare phrase-based methods to PSCFG methods.

Ur.En. Conf.	# Sentences	# Src, Tgt words	# 1..5Grams (mono.)
TARGET	207K	2.2M(ur) 2.1M(en)	4M

Table A.5: Urdu-to-English Data configurations under which we compare phrase-based methods to PSCFG methods.

DataSet	# Sentences	NumRefs	Average # Words per Reference
MT02	878	4	30
MT03	919	4	31
MT04 (dev)	1788	4	32
MT05	1082	4	32
MT06	1554	4	28
MT08	1357	4	30
UrEn-dev	1840	1	21
UrEn-MT08	1864	4	22

Table A.6: NIST development data statistics for Chinese-to-English and Arabic–to-English (MT-0x) and Urdu-to-English (UrEn-x). For each data set we note the number of sentences, the number of source words, number of references per source sentence, and the average number of words per reference.

Bibliography

Alfred V. Aho and Jeffrey D. Ullmann. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3, 1969.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the Annual Meeting of the Association of Computational Linguistics (ACL)*, 2005.

Adam L. Berger, Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Andrew S. Kehler, and Robert L. Mercer. Language translation apparatus and method using context-based translation models. *United States Patent 5510981*, 1996a.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22 (1), 1996b.

Phil Blunsom, Trevor Cohn, and Miles Osborne. A discriminative latent variable model for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2007.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in MT. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2), 1990.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2), 1993.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. Meta-Evaluation of machine translation. In *Proceedings of the Workshop on Statistical Machine Translation, ACL*, 2007.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. Findings of the 2009 workshop on statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation, European Association of Computational Linguistics (EACL)*, 2009.

Francisco Casacuberta and Colin de la Higuera. Computational complexity of problems on probabilistic grammars and transducers. In *Proceedings of the International Colloquium on Grammatical Inference (ICGI)*, 2000.

Eugene Charniak. A maximum entropy-inspired parser. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, 2000.

Eugene Charniak and Mark Johnson. Coarse-to-fine-grained N-best parsing and discriminative reranking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moorea, Michael Pozar, and Theresa Vu. Multilevel coarse-to-fine PCFG parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, 2006.

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

David Chiang. Hierarchical phrase based translation. *Computational Linguistics*, 33 (2), 2007.

Michael Collins. *Head-driven Statistical Models for Natural Language Processing*. PhD thesis, University of Pennsylvania, 1999.

Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 2005.

Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, 1985.

A.P Dempster, N.M Laird, and D.B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Society*, B(39), 1977.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *In Proceedings ARPA Workshop on Human Language Technology*, 2002.

Heidi J. Fox. Phrasal cohesion and statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

Michael Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, 2004.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. Scalable inferences and training of context-rich syntax translation models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.

Giorgio Gallo, Giustino Longo, Stefano Pallottino, and San Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2), 1993.

Bryant Huang and Kevin Knight. Relabeling syntax trees to improve syntax-based machine translation quality. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, 2006.

Liang Huang and David Chiang. Better K-best parsing. In *Proceedings of the International Workshop on Parsing Technologies (IWPT)*, 2005.

Liang Huang and David Chiang. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2007.

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. Evaluating translational correspondance using annotation projection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

T. Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, Air Force Cambridge Research Lab, 1965.

Dan Klein and Christoper Manning. Accurate unlexicalized parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2003.

Dan Klein and Christopher D. Manning. Parsing and hypergraphs. In *In Proceedings of the International Workshop on Parsing Technologies (IWPT)*, 2001.

R. Kneser and Hermann Ney. Improved backing-off for n-gram language modeling. In *In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1995.

Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics, Squibs and Discussion*, 25(4), 1999.

Kevin Knight and Yasser Al-Onaizan. Translation with finite-state devices. In *Proceedings of the Association for Machine Translation in the Americas (AMTA)*, 1998.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the Machine Translation Summit*, 2005.

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2007.

Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, 2003.

Philipp Koehn, Franz J. Och, and Daniel Marcu. Pharaoh: A beam search decoder for phrase-base statistical machine translation models. In *Proceedings of the Association for Machine Translation in the Americas (AMTA)*, 2004.

Phillip Koehn and Christof Monz. Shared task: Statistical machine translation between european languages. In *Proceedings of the Workshop on Parallel Corpora, ACL*, 2005.

Shankar Kumar and William Byrne. Minimum bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, Boston,MA, May 27-June 1 2004.

Zhifei Li and Sanjeev Khudanpur. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of the Syntax and Structure in Statistical Translation Workshop ACL*, 2008.

Adam Lopez. Statistical machine translation. *ACM Computing Surveys*, 40(3), 2008.

Lidia Mangu, Eric Brill, and Andreas Stolke. Finding consensus among words: Lattice-based word error minimization. In *Proceeding of the European Conference on Speech Comunication and Technology (EUROSPEECH)*, 1999.

Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. SPMT: Statistical machine translation with syntactified target language phrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sydney, Australia, 2006.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2), 1993.

Takuya Matsusaki, Yusuke Miyao, , and Junichi Tsujii. Probabilistic CFG with latent annotations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

Jonathan May and Kevin Knight. A better N-best list: Practical determinization of weighted finite tree automata. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, 2006.

I. Dan Melamed. Multitext grammars and synchronous parsers. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, 2003.

Sonja Nießen and Herman Ney. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2), 2004.

Franz J. Och. Minimum error rate training in statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2003.

Franz J. Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

Franz J. Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4), 2004.

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

Franz J. Och, Christoph Tillmann, and Hermann Ney. Improved alignment models for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*, 1999.

Franz J. Och, Daniel Gildea, Sanjeev Khudanpur, and Anoop Sarkar. A smorgasbord of features for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, 2004.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

Michael Paul. Overview of the IWSLT 2006 evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, 2006.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.

Slav Petrov, Aria Haghighi, and Dan Klein. Coarse-to-fine syntactic machine translation using language projections. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2008.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognitions. *Proceedings of the IEEE*, 77(2), 1989.

Remko Scha. Taaltheorie en taaltechnologie; competence en performance (language theory and language technology; competence and performance). *Computertoepassingen in de Neerlandistiek*, 1990.

Stuart Shieber, Yves Schabes, and Fernando Pereira. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–15, 1995.

Khalil Sima'an. Computational complexity of probabilistic disambiguation. *Grammars*, 2002.

David A. Smith and Noah A. Smith. Probabilistic models of nonprojective dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2007.

Mark Steedman. Alternative quantifier scope in CCG. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 1999.

Christoph Tillmann, Stephan Vogel, Hermann Ney, and A. Zubiaga. A DP based search using monotone alignments in statistical translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 1997.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. Lattice minimum bayes-risk decoding for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2008.

Ashish Venugopal, Stephan Vogel, and Alex Waibel. Effective phrase translation extraction from alignment models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2003.

Ashish Venugopal, Andreas Zollmann, and Stephan Vogel. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, 2007.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. Preference grammars: Softening syntactic constraints to improve statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, 2009.

Enrique Vidal. Finite-state speech-to-speech translation. In *In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1997.

Stephan Vogel. PESA: Phrase pair extraction as sentence splitting. In *Proceedings of the Machine Translation Summit*, 2005.

Stephan Vogel, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, and Alex Waibel. The CMU statistical translation system. In *Proceedings of Machine Translation Summit*, 2003.

Chao Wang, Michael Collins, and Philipp Koehn. Chinese syntactic reordering for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2007.

Ye-Yi Wang and Alex Waibel. Decoding algorithm in statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 1997.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. Left-to-right target generation for hierarchical phrase based translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.

John S. White, Theresa O'Connell, and Francis O'Mara. The ARPA MT evaluation methodologies: Evolution, lessons, and future approaches. In *Proceedings of the Association for Machine Translation in the Americas (AMTA)*, 1994.

Dekai Wu. A polynomial time algorithm for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 1996.

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3), 1997.

Fei Xia and Michael McCord. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of the Conference on Computational Linguistics (COLING)*, 2004.

Richard Zens and Hermann Ney. Discriminative reordering models for statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation, HLT/NAACL*, 2006a.

Richard Zens and Hermann Ney. N-gram posterior posterior probabilities for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*, 2006b.

Hao Zhang and Daniel Gildea. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

Hao Zhang and Daniel Gildea. Efficient multi-pass decoding for synchronous context free grammars. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2008.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology*

Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL), 2006.

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

Andreas Zollmann, Ashish Venugopal, Franz J. Och, and Jay Ponte. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of the Conference on Computational Linguistics (COLING)*, 2008.