An Efficient Two-Pass Approach to Synchronous-CFG Driven Statistical MT

Ashish Venugopal and **Andreas Zollmann** and **Stephan Vogel**
School of Computer Science, Carnegie Mellon University, Pittsburgh
interACT Lab, University of Karlsruhe
{ashishv,zollmann,vogel+}@cs.cmu.edu

Abstract

We present an efficient, novel two-pass approach to mitigate the computational impact resulting from online intersection of an n-gram language model (LM) and a probabilistic synchronous context-free grammar (PSCFG) for statistical machine translation. In first pass CYK-style decoding, we consider first-best chart item approximations, generating a hypergraph of sentence spanning target language derivations. In the second stage, we instantiate specific alternative derivations from this hypergraph, using the LM to *drive* this search process, recovering from search errors made in the first pass. Model search errors in our approach are comparable to those made by the state-of-the-art "Cube Pruning" approach in (Chiang, 2007) under comparable pruning conditions evaluated on both hierarchical and syntax-based grammars.

1 Introduction

Syntax-driven (Galley et al., 2006) and hierarchical translation models (Chiang, 2005) take advantage of probabilistic synchronous context free grammars (PSCFGs) to represent structured, lexical reordering constraints during the decoding process. These models extend the domain of locality (over phrase-based models) during decoding, representing a significantly larger search space of possible translation derivations. While PSCFG models are often induced with the goal of producing grammatically correct target translations as an implicit syntax-structured language model, we acknowledge the value of n-gram language models (LM) in phrase-based approaches.

Integrating n-gram LMs into PSCFGs based decoding can be viewed as online intersection of the PSCFG grammar with the finite state machine represented by the n-gram LM, dramatically increasing the effective number of nonterminals in the decoding grammar, rendering the decoding process essentially infeasible without severe, beam-based lossy pruning. The alternative, simply decoding without the n-gram LM and rescoring N-best alternative translations, results in substantially more search errors, as shown in (Zollmann and Venugopal, 2006).

Our two-pass approach involves fast, approximate synchronous parsing in a first stage, followed by a second, detailed exploration through the resulting hypergraph of sentence spanning derivations, using the n-gram LM to drive that search. This achieves search errors comparable to a strong "Cube Pruning" (Chiang, 2007), single-pass baseline. The first pass corresponds to a severe parameterization of Cube Pruning considering only the first-best (LM integrated) chart item in each cell while maintaining unexplored alternatives for second-pass consideration. Our second stage allows the integration of long distance and flexible history n-gram LMs to drive the search process, rather than simply using such models for hypothesis rescoring.

We begin by discussing the PSCFG model for statistical machine translation, motivating the need for effective n-gram LM integration during decoding. We then present our two-pass approach and discuss Cube Pruning as a state-of-the-art baseline. We present results in the form of search error analysis and translation quality as measured by the BLEU score (Papineni et al., 2002) on the IWSLT 06 text translation task (Eck and Hori, 2005)[1], comparing Cube Pruning with our two-pass approach.

2 Synchronous Parsing for SMT

Probabilistic Synchronous Context Free Grammar (PSCFG) approaches to statistical machine translation use a source terminal set (source vocabulary) $\mathcal{T}_S$, a target terminal set (target vocabulary) $\mathcal{T}_T$ and a shared nonterminal set $\mathcal{N}$ and induce rules of the form

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

where (i) $X \in \mathcal{N}$ is a nonterminal, (ii) $\gamma \in (\mathcal{N} \cup \mathcal{T}_S)^*$ is a sequence of nonterminals and source terminals, (iii) $\alpha \in (\mathcal{N} \cup \mathcal{T}_T)^*$ is a sequence of nonterminals and target terminals, (iv) the number $cnt(\gamma)$ of nonterminal occurrences in γ is equal to the number $cnt(\alpha)$ of nonterminal occurrences in α, (v) $\sim$: $\{1, \ldots, cnt(\gamma)\} \rightarrow \{1, \ldots, cnt(\alpha)\}$ is a one-to-one mapping from nonterminal occurrences in γ to nonterminal occurrences in α, and (vi) $w \in [0, \infty)$ is a non-negative real-valued weight assigned to the rule. We will assume $\sim$ to be implicitly defined by indexing the NT occurrences in γ from left to right starting with 1, and by indexing the NT occurrences in α by the indices of their corresponding counterparts in γ. Syntax-oriented PSCFG approaches typically ignore source structure, instead focussing on generating syntactically well formed target derivations. (Galley et al., 2006) use syntactic constituents for the PSCFG nonterminal set and (Zollmann and Venugopal, 2006) take advantage of CCG (Steedman, 1999) categories, while (Chiang, 2005) uses a single generic nonterminal. PSCFG derivations function analogously to CFG derivations. Given a source sentence f, the translation task under a PSCFG grammar can be expressed as

$$\hat{e} = \underset{\{e \mid \exists D.\ \mathrm{src}(D)=f, \mathrm{tgt}(D)=e\}}{\arg\max} P(D)$$

where $\mathrm{tgt}(D)$ refers to the target terminal symbols generated by the derivation D and $\mathrm{src}(D)$ refers to the source terminal symbols spanned by D. The score (also laxly called probability, since we never need to compute the partition function) of a derivation D under a log-linear model, referring to the rules r used in D, is:

$$P(D) = \frac{1}{Z} P_{LM}(\mathrm{tgt}(D))^{\lambda_{LM}} \times \prod_i \prod_{r \in D} \phi_i(r)^{\lambda_i}$$

where ϕ_i refers to features defined on each rule, and P_{LM} is a g-gram LM probability applied to the target terminal symbols generated by the derivation D. Introducing the LM feature defines dependencies across adjacent rules used in each derivation, and requires modifications to the decoding strategy. Viewing the LM as a finite-state machine, the decoding process involves performing an intersection between the PSCFG grammar and the g-gram LM (Bar-Hillel et al., 1964). We present our work under the construction in (Wu, 1996), following notation from (Chiang, 2007), extending the formal description to reflect grammars with an arbitrary number of nonterminals in each rule.

2.1 Decoding Strategies

In Figure 1, we reproduce the decoding algorithm from (Chiang, 2007) that applies a PSCFG to translate a source sentence in the same notation (as a deductive proof system (Shieber et al., 1995)), generalized to handle more than two non-terminal pairs. Chart items $[X, i, j, e] : w$ span $j - i$ words in the source sentence $f_1 \cdots f_n$, starting at position $i + 1$, and have weight w (equivalent to $P(D)$), and $e \in (\mathcal{T}_T \cup \{\star\})^*$ is a sequence of target terminals, with possible elided parts, marked by $\star$. Functions p, q whose domain is $\mathcal{T}_T \cup \{\star\}$ are defined in (Chiang, 2007) and are repeated here for clarity.

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

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

The function q elides elements from a target language terminal sequence, leaving the leftmost and rightmost $g - 1$ words, replacing the elided words

[1] While IWSLT represents a limited resource translation task (120K sentences of training data for Chinese-English), the problem of efficient n-gram LM integration is still critically important to efficient decoding, and our contributions can be expected to have an even more significant impact when decoding with grammars induced from larger corpora.

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

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

$$\frac{Z \rightarrow \langle f_{i+1}^{i_1} (X^1)_1 f_{j_1+1}^{i_2} \cdots (X^{m-1})_{m-1} f_{j_{m-1}+1}^{i_m} (X^m)_m f_{j_m+1}^{j}, \alpha \rangle : w \quad \left[X^1, i_1, j_1; e_1\right] : w_1 \ \cdots \ [X^m, i_m, j_m; e_m] : w_m}{[Z, i, j, q(\alpha')] : ww_1 \cdots w_m p(\alpha') \quad (\text{where } \alpha' = \alpha \left[e_1/(X^1)_1, \ldots, e_m/(X^m)_m\right])}$$

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

Figure 1. CYK parsing with integrated g-gram LM. The inference rules are explored in ascending order of $j - i$. Here $\alpha\,[e/Y_i]$ is the string α where the NT occurrence Y_i is replaced by e. Functions q and p are explained in the text.

with a single $\star$ symbol. The function p returns g-gram LM probabilities for target terminal sequences, where the $\star$ delineates context boundaries, preventing the calculation from spanning this boundary. We add a distinguished start nonterminal S to generate sentences spanning target translations beginning with $g-1$ $\langle s \rangle$ symbols and ending with $g-1$ $\langle \backslash s \rangle$ symbols. This can e.g. be achieved by adding for each nonterminal X a PSCFG rule

$$S \rightarrow \langle X, \langle s \rangle^{g-1} X \langle \backslash s \rangle^{g-1}, 1 \rangle$$

We are only searching for the derivation of highest probability, so we can discard identical chart items that have lower weight. Since chart items are defined by their left-hand side nonterminal production, span, *and* the LM contexts e, we can safely discard these identical items since q has retained all context that could possibly impact the LM calculation. This process is commonly referred to as item recombination. Backpointers to antecedent cells are typically retained to allow N-Best extraction using an algorithm such as (Huang and Chiang, 2005).

The impact of g-gram LM intersection during decoding is apparent in the final deduction step. Generating the set of consequent Z chart items involves combining m previously produced chart cells. Since each of these chart cells with given source span $[i, j]$ is identified by nonterminal symbol X and LM context e, we have at worst $|\mathcal{N}| * |\mathcal{T}_T|^{2(g-1)}$ such chart cells in a span. The runtime of this algorithm is thus

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

where K is the maximum number of NT pairs per rule and n the source sentence length. Without severe pruning, this runtime is prohibitive for even the smallest induced grammars. Traditional pruning approaches that limit the number of consequents after they are produced are not effective since they first require that the cost of each consequent be computed (which requires calls to the g-gram LM).

Restrictions to the grammar afford alternative decoding strategies to reduce the runtime cost of synchronous parsing. (Zhang et al., 2006) "binarize" grammars into CNF normal form, while (Watanabe et al., 2006) allow only Griebach-Normal form grammars. (Wellington et al., 2006) argue that these restrictions reduce our ability to model translation equivalence effectively. We take an agnostic view on the issue; directly addressing the question of efficient LM intersection rather than grammar construction.

3 Two-pass LM Intersection

We propose a two-pass solution to the problem of online g-gram LM intersection. A naive two-pass approach would simply ignore the LM interactions during parsing, extract a set of N derivations from the sentence spanning hypergraph and rescore these derivations with the g-gram LM. In practice, this approach performs poorly (Chiang, 2007; Zollmann and Venugopal, 2006). While parsing time is dramatically reduced (and N-best extraction time is negligible), N is typically significantly less than the complete number of possible derivations and substantial search errors remain. We propose an approach that builds upon the concept of a second pass but uses the g-gram LM to search for alternative, better translations.

3.1 First pass: parsing

We begin by relaxing the criterion that determines when two chart items are equivalent during parsing. We consider two chart items to be equivalent (and therefore candidates for recombination) if they have matching left-hand side nonterminals, and span. We no longer require them to have the same LM context e. We *do* however propagate the e, w for the chart item with highest score, causing the algorithm to still compute LM probabilities during parsing. As a point of notation, we refer to such a chart item by annotating its e, w as e^1, w^1, and we refer to them as approximate items (since they have made a first-best approximation for the purposes of LM calculation). These approximate items labeled with e^1, w^1 are used in consequent parse calculations.

The parsing algorithm under this approximation stays unchanged, but parsing time is dramatically reduced. The runtime complexity of this algorithm is now $\mathcal{O}\left(n^3|\mathcal{N}|^K\right)$ at the cost of significant search errors (since we ignored most LM contexts that we encountered).

This relaxation is different from approaches that do not use the LM during parsing. The sentence spanning item *does* have LM probabilities associated with it (but potentially valuable chart items were not considered during parsing). Like in traditional parsing, we retain the recombined items in the cell to allow us to explore new derivations in a second stage.

3.2 Second pass: hypergraph search

The goal item of the parsing step represents a sentence spanning hypergraph of alternative derivations. Exploring alternatives from this hypergraph and updating LM probabilities can now reveal derivations with *higher* scores that were not considered in the first pass. Exploring the whole space of alternative derivations in this hypergraph is clearly infeasible. We propose a g-gram LM driven heuristic search "H.Search" of this space that allows the g-gram LM to decide which section of the hypergraph to explore. By construction, traversing a particular derivation item from the parse chart in target-side left-to-right, depth-first order yields the correctly ordered sequence of target terminals that is the translation represented by this item. Now consider a *partial* traversal of the item in that order, where we generate only the first M target terminals, leaving the rest of the item in its original backpointer form. We informally define our second pass algorithm based on these partial derivation items.

Consider a simple example, where we have parsed a source sentence, and arrived at a sentence spanning item obtained from a rule with the following target side:

$$\text{NP}_2\ \text{VP}_3\ \text{PP}_1$$

and that the item's best-score estimate is w. A partial traversal of this item would replace NP_2 with one of the translations available in the chart cell underlying NP_2 (called "unwinding"), and recalculate the weights associated with this item, taking into account the alternative target terminals. Assuming "the nice man" was the target side of the best scoring item in NP_2, the respective traversal would maintain the same weight. An alternative item at NP_2 might yield "a nice man". This partial traversal represents a possible item that we did not consider during parsing, and recalculating LM probabilities for this new item (based on approximate items VP_3 and PP_1) yields weight w_2:

$$\text{the nice man VP}_3\ \text{PP}_1 : w_1 = w$$

$$\text{a nice man VP}_3\ \text{PP}_1 : w_2$$

Alternative derivation items that obtain a higher score than the best-score estimates represent recovery from search errors. Our algorithm is based on the premise that these items should be traversed further, with the LM continuing to score newly generated target words. These partially traversed items are placed on an agenda (sorted by score). At each step of the second pass search, we select those items from the agenda that are within a search beam of Z from the best item, and perform the unwind operation on each of these items. Since we unwind partial items from left-to-right the g-gram LM is able to influence the search through the space of alternative derivations.

Applying the g-gram LM on partial items with leading only-terminal symbols allows the integration of high- / flexible-order LMs during this second stage process, and has the advantage of exploring only those alternatives that participate in sentence spanning, high scoring (considering both LM and translation model scores) derivations. While we do not evaluate such models here, we note that H.Search was developed specifically for the integration of such models during search.

We further note that partial items that have generated translations that differ only in the word positions up to $g - 1$ words before the first nonterminal site can be recombined (for the same reasons as during LM intersected parsing). For example, when considering a 3-gram LM, the two partial items above can be recombined into one equivalence class, since partial item LM costs resulting from these items would only depend on 'nice man', but not on 'a' vs. 'the'. Even if two partial items are candidates for recombination due to their terminal words, they must also have identical backpointers (representing a set of approximate parse decisions for the rest of the sentence, in our example VP$_3$PP$_1$). Items that are filed into existing equivalence classes with a lower score are not put onto the agenda, while those that are better, or have created new equivalence classes are scheduled onto the agenda. For each newly created partial derivation, we also add a backpointer to the "parent" partial derivation that was unwound to create it.

This equivalence classing operation transforms the original left-hand side NT based hypergraph into an (ordinary) graph of partial items. Each equivalence class is a node in this new graph, and recombined items are the edges. Thus, N-best extraction can now be performed on this graph. We use the extraction method from (Huang and Chiang, 2005).

The expensive portion of our algorithm lies in the unwinding step, in which we generate a new partial item for each alternative at the non-terminal site that we are "unwinding". For each new partial item, we factor *out* LM estimates and rule weights that were used to score the parent item, and factor *in* the LM probabilities and rule weights of the alternative choice that we are considering. In addition, we must also update the new item's LM estimates for the remaining non-terminal and terminal symbols that depend on this new left context of terminals. Fortunately, the number of LM calculations per new item is constant, i.e., does not dependent on the length of the partial derivation, or how unwound it is. Only $(g - 1) * 2$ LM probabilities have to be re-evaluated per partial item. We now define this "unwind-recombine" algorithm formally.

3.2.1 The unwind-recombine algorithm

Going back to the first-pass parsing algorithm (Figure 1), remember that each application of a grammar rule containing nonterminals corresponds to an application of the third inference rule of the algorithm. We can assign chart items C created by the third inference rule a *back-pointer (BP) target side* as follows: When applying the third inference rule, each nonterminal occurrence $(X^k)_k$ in the corresponding $Z \rightarrow \langle \lambda, \alpha \rangle$ grammar rule corresponds to a chart cell $[X^k, i_k, j_k]$ used as an antecedent for the inference rule. We assign a BP target side for C by replacing NT occurrences in α (from the rule that created C) with backpointers to their corresponding antecedent chart cells. Further we define the distinguished backpointer P_S as the pointer to the goal cell $[S, 0, n] : w^*$.

The deductive program for our second-pass algorithm is presented in Figure 2. It makes use of two kind of items. The first, $\{P \rightarrow \alpha; e^1\} : w$, links a backpointer P to a BP target side, storing current-item vs. best-item correction terms in form of an LM context e^1 and a relative score w. The second item form $[[e; \alpha]]$ in this algorithm corresponds to partial left-to-right traversal states as described above, where e is the LM context of the traversed and unwound translation part, and α the part that is yet to be traversed and whose backpointers are still to be unwound. The first deduction rule presents the logical axioms, creating BP items for each backpointer used in a NT inference rule application during the first-pass parsing step. The second deduction rule represents the unwinding step as discussed in the example above. These deductions govern a search for derivations through the hypergraph that is driven by updates of rule weights and LM probabilities when unwinding non-first-best hypotheses. The functions p and q are as defined in Section 2, except that the domain of q is extended to BP target sides by first replacing each back-pointer with its corresponding chart cell's LM context and then applying the original q on the resulting sequence of target-terminals and $\star$ symbols.[2] Note that w', which was computed by the first deduction rule, adjusts the current hypothesis' weight

[2] Note also that $p(\langle s \rangle^{g-1} \star \langle \backslash s \rangle^{g-1}) = 1$ as the product over the empty set is one.

$$\frac{}{\{P \rightarrow \alpha; e^1\} : w'/w} \quad (P \text{ back-points to 1st-pass cell } [X, i, j; e^1] : w;\ \alpha \text{ and } w' \text{ are BP-target-side and weight of one of that cell's items})$$

$$\frac{[[e; P\alpha_{end}]] : w \quad \{P \rightarrow \alpha_{lex}\alpha_{mid}; e^1\} : w'}{[[q(e\alpha_{lex}); \alpha_{mid}\alpha_{end}]] : ww'p[eq(\alpha_{lex}\alpha_{mid})]p[q(e\alpha_{lex}\alpha_{mid})q(\alpha_{end})]/p(ee^1)/p[q(ee^1)q(\alpha_{end})]} \begin{pmatrix} \alpha_{lex} \text{ contains no BPs and} \\ \alpha_{mid} = P'\alpha' \text{ or } \alpha_{mid} = \varepsilon \end{pmatrix}$$

Figure 2. Left-to-right LM driven hypergraph search of the sentence spanning hypergraph; ε denotes the empty word. Non-logical (*Start*) axiom: $[[\varepsilon; P_S]] : w^*$; Goal item: $[[\langle s \rangle^{g-1} \star \langle \backslash s \rangle^{g-1}; \varepsilon]] : w$

that is based on the first-best instance of P to the actually chosen instance's weight. Further, the ratio $p(eq(\alpha_{lex}\alpha_{mid}))/p(ee^1)$ adjusts the LM probabilities of P's instantiation given its left context, and $p[q(e\alpha_{lex}\alpha_{mid})q(\alpha_{end})]/p[q(ee^1)q(\alpha_{end})]$ adjusts the LM probabilities of the $g - 1$ words right of P.

4 Alternative Approaches

We evaluate our two pass hypergraph search "H.Search" against the strong single pass Cube Pruning (CP) baseline as mentioned in (Chiang, 2005) and detailed in (Chiang, 2007). In the latter work, the author shows that CP clearly outperforms both the naive single pass solution of severe pruning as well as the naive two-pass rescoring approach. Thus, we focus on comparing our approach to CP.

CP is an optimization to the intersected LM parsing algorithm presented in Figure 1. It addresses the creation of the $\prod_{k=1}^{K} \mid [X_k, i_k, j_k, *] \mid$ chart items when generating consequent items. CP amounts to an early termination condition when generating the set of possible consequents. Instead of generating all consequents, and then pruning away the poor performers, CP uses the K-Best extraction approach of (Huang and Chiang, 2005) to select the best K consequents only, at the cost of potential search errors. CP's termination condition can be defined in terms of an absolute number of consequents to generate, or by terminating the generation process when a newly generated item is worse (by β) than the current best item for the same left-hand side and span. To simulate comparable pruning criteria we parameterize each method with soft-threshold based criteria only (β for CP and Z for H.Search) since counter based limits like K have different effects in CP (selecting e labeled items) vs H.Search (selecting rules since items are not labeled with e).

5 Experimental Framework

We present results on the IWSLT 2006 Chinese to English translation task, based on the Full BTEC corpus of travel expressions with 120K parallel sentences (906K source words and 1.2m target words). The evaluation test set contains 500 sentences with an average length of 10.3 source words.

Grammar rules were induced with the syntax-based SMT system "SAMT" described in (Zollmann and Venugopal, 2006), which requires initial phrase alignments that we generated with "GIZA++" (Koehn et al., 2003), and syntactic parse trees of the target training sentences, generated by the Stanford Parser (D. Klein, 2003) pre-trained on the Penn Treebank. All these systems are freely available on the web.

We experiment with 2 grammars, one syntax-based (3688 nonterminals, 0.3m rules), and one purely hierarchical (1 generic nonterminal, 0.05m rules) as in (Chiang, 2005). The large number of nonterminals in the syntax based systems is due to the CCG extension over the original 75 Penn Treebank nonterminals. Parameters λ used to calculate $P(D)$ are trained using MER training (Och, 2003) on development data.

6 Comparison of Approaches

We evaluate each approach by considering both search errors made on the development data for a fixed set of model parameters, and the BLEU metric to judge translation quality.

6.1 Search Error Analysis

While it is common to evaluate MT quality using the BLEU score, we would like to evaluate search errors made as a function of "effort" made by each algorithm to produce a first-best translation. We consider two metrics of effort made by each algorithm.

We first evaluate search errors as a function of novel queries made to the g-gram LM (since LM calls tend to be the dominant component of runtime in large MT systems). We consider novel queries as those that have not already been queried for a particular sentence, since the repeated calls are typically efficiently cached in memory and do not affect runtime significantly. Our goal is to develop techniques that can achieve low search error with the fewest novel queries to the g-gram LM.

To appreciate the practical impact of each algorithm, we also measure search errors as a function of the number of seconds required to translate a fixed unseen test set. This second metric is more sensitive to implementation and, as it turned out, even compiler memory management decisions.

We define search errors based on the weight of the best sentence spanning item. Treating weights as negative log probabilities (costs), we accumulate the value of the lowest cost derivation for each sentence in the testing data as we vary pruning settings appropriate to each method. Search errors are reduced when we are able to lower this accumulated model cost. We prefer approaches that yield low model cost with the least number of LM calls or number of seconds spent decoding.

It is important to note that model cost $(-\log(P(D)))$ is a function of the parameters λ which have been trained using MER training. The parameters used for these experiments were trained with the CP approach; in practice we find that either approach is effective for MER training.

6.2 Results

Figure 3 and Figure 4 plot model cost as a function of LM cache misses for the IWSLT Hierarchical and Syntax based systems, while Figure 5 plots decoding time. The plots are based on accumulated model cost, decoding time and LM cache misses over the IWSLT Test 06 set. For H.Search, we vary the beam parameter Z for a fixed value of $\beta = 5$ during parsing while for CP, we vary β. We also limit the total number of items on the agenda at any time to 1000 for H.Search as a memory consideration. We plot each method until we see no change in BLEU score for that method. BLEU scores for each parameter setting are also noted on the plots.

For both the hierarchical and syntax based grammars we see that the H.Search method 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.[3] While the search beam of Z mitigates the impact of the estimated scores during H.Search's second pass, the score is still not an admissible heuristic for error-free search. We suspect that simple methods to "underestimate" the score of a partial derivation's remaining nonterminals could bridge this gap in search error. BLEU scores in the regions of lowest model cost tend to be reasonably stable and reflect comparable translation performance for both methods.

Under both H.Search and CP, the hierarchical grammar ultimately achieves a BLEU score of 19.1%, while the syntactic grammar's score is approximately 1.5 points higher at 20.7%. The hierarchical grammar demonstrates a greater variance of BLEU score for both CP and H.Search compared to the syntax-based grammar. The use of syntactic structure serves as an additional model of target language fluency, and can explain the fact that syntax based translation quality is more robust to differences in the number of g-gram LM options explored.

Decoding time plots shows a similar result, but with diminished relative improvement for the H.Search method. Profiling analysis of the H.Search method shows that significant time is spent simply on allocating and deallocating memory for partial derivations on top of the scoring times for these items. We expect to be able to reduce this overhead significantly in future work.

7 Conclusion

We presented an novel two-pass decoding approach for PSCFG-based machine translation that achieves search errors comparable to the state of the art Cube Pruning method. By maintaining comparable, sentence spanning derivations we allow easy integration of high or flexible order LMs as well as sentence level syntactic features during the search process. We plan to evaluate the impact of these more powerful models in future work. We also hope to address the question of how much search error is tolerable to

[3] Analysis of *total* LM calls made by each method (not presented here) shows the H.Search makes significantly fewer (1/2) total LM calls than CP to achieve each model cost.

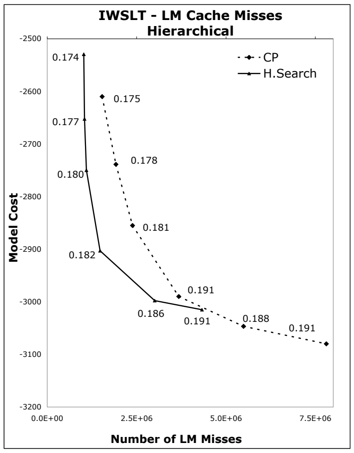

Figure 3. LM caches misses for IWSLT hierarchical grammar and BLEU scores for varied pruning parameters

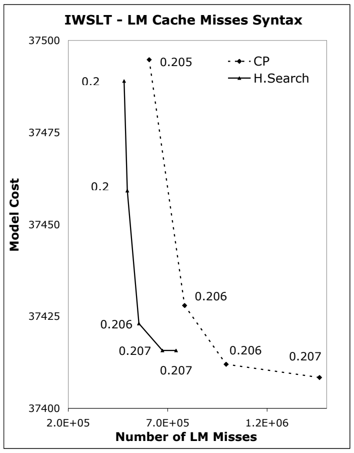

Figure 4. LM caches misses for IWSLT syntactic grammar and BLEU scores for varied pruning parameters

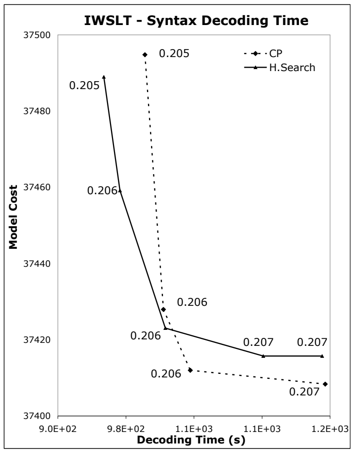

Figure 5. Decoding time (s) for IWSLT syntactic grammar and BLEU scores for varied pruning parameters

run MER training and still generate parameters that generalize well to test data. This point is particularly relevant to evaluate the use of search error analysis.

Acknowledgments This work has been funded by the European Union under the integrated project TC-STAR - Technology and Corpora for Speech to Speech Translation (IST-2002-FP6-506738, http://www.tc-star.org).

References

Bar-Hillel, M.Perles, and E.Shamir. 1964. An efficient context-free parsing algorithm. *Communications of the Assocation for Computing Machinery*.

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

David Chiang. 2007. Hierarchical phrase based translation. *To Appear in the Journal of Computational Linguistics*.

C. Manning D. Klein. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*.

Matthias Eck and Chiori Hori. 2005. Overview of the IWSLT 2005 evaluation campaign. In *Proc. of International Workshop on Spoken Language Translation*, pages 11–17.

Michael Galley, M. Hopkins, Kevin Knight, and Daniel Marcu. 2006. Scalable inferences and training of context-rich syntax translation models. In *Proc. of NAACL-HLT*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proc. of the 9th International Workshop on Parsing Technologies*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT/NAACL*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL*, Sapporo, Japan, July 6-7.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

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

Mark Steedman. 1999. Alternative quantifier scope in CCG. In *Proc. of ACL*.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase based translation. In *ACL*.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *ACL*.

Dekai Wu. 1996. A polynomial time algorithm for statistical machine translation. In *Proc. of the Association for Computational Linguistics*.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proc. of HLT/NAACL*.

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