Efficient Multi-pass Decoding for Synchronous Context Free Grammars

Hao Zhang and **Daniel Gildea**
Computer Science Department
University of Rochester
Rochester, NY 14627

Abstract

We take a multi-pass approach to machine translation decoding when using synchronous context-free grammars as the translation model and n-gram language models: the first pass uses a bigram language model, and the resulting parse forest is used in the second pass to guide search with a trigram language model. The trigram pass closes most of the performance gap between a bigram decoder and a much slower trigram decoder, but takes time that is insignificant in comparison to the bigram pass. An additional fast decoding pass maximizing the expected count of correct translation hypotheses increases the BLEU score significantly.

1 Introduction

Statistical machine translation systems based on synchronous grammars have recently shown great promise, but one stumbling block to their widespread adoption is that the decoding, or search, problem during translation is more computationally demanding than in phrase-based systems. This complexity arises from the interaction of the tree-based translation model with an n-gram language model. Use of longer n-grams improves translation results, but exacerbates this interaction. In this paper, we present three techniques for attacking this problem in order to obtain fast, high-quality decoders.

First, we present a two-pass decoding algorithm, in which the first pass explores states resulting from an integrated bigram language model, and the second pass expands these states into trigram-based states. The general bigram-to-trigram technique is common in speech recognition (Murveit et al., 1993), where lattices from a bigram-based decoder are re-scored with a trigram language model. We examine the question of whether, given the reordering inherent in the machine translation problem, lower order n-grams will provide as valuable a search heuristic as they do for speech recognition.

Second, we explore heuristics for agenda-based search, and present a heuristic for our second pass that combines precomputed language model information with information derived from the first pass. With this heuristic, we achieve the same BLEU scores and model cost as a trigram decoder with essentially the same speed as a bigram decoder.

Third, given the significant speedup in the agenda-based trigram decoding pass, we can rescore the trigram forest to maximize the expected count of correct synchronous constituents of the model, using the product of inside and outside probabilities. Maximizing the expected count of synchronous constituents approximately maximizes BLEU. We find a significant increase in BLEU in the experiments, with minimal additional time.

2 Language Model Integrated Decoding for SCFG

We begin by introducing Synchronous Context Free Grammars and their decoding algorithms when an n-gram language model is integrated into the grammatical search space.

A **synchronous CFG** (SCFG) is a set of context-free rewriting rules for recursively generating string pairs. Each synchronous rule is a pair of CFG rules

Proceedings of ACL-08: HLT, pages 209–217,
Columbus, Ohio, USA, June 2008.

with the nonterminals on the right hand side of one CFG rule being one-to-one mapped to the other CFG rule via a permutation π. We adopt the SCFG notation of Satta and Peserico (2005). Superscript *indices* in the right-hand side of grammar rules:

$$X \rightarrow X_1^{(1)}...X_n^{(n)},\ X_{\pi(1)}^{(\pi(1))}...X_{\pi(n)}^{(\pi(n))}$$

indicate that the nonterminals with the same index are linked across the two languages, and will eventually be rewritten by the same rule application. Each X_i is a variable which can take the value of any nonterminal in the grammar.

In this paper, we focus on binary SCFGs and without loss of generality assume that only the preterminal unary rules can generate terminal string pairs. Thus, we are focusing on Inversion Transduction Grammars (Wu, 1997) which are an important subclass of SCFG. Formally, the rules in our grammar include preterminal unary rules:

$$X \rightarrow \mathbf{e}/\mathbf{f}$$

for pairing up words or phrases in the two languages and binary production rules with straight or inverted orders that are responsible for building up upper-level synchronous structures. They are straight rules written:

$$X \rightarrow [YZ]$$

and inverted rules written:

$$X \rightarrow \langle YZ \rangle.$$

Most practical non-binary SCFGs can be binarized using the synchronous binarization technique by Zhang et al. (2006). The Hiero-style rules of (Chiang, 2005), which are not strictly binary but binary only on nonterminals:

$$X \rightarrow \text{ yu } X^{(1)} \text{ you } X^{(2)};\ \text{have } X^{(2)} \text{ with } X^{(1)}$$

can be handled similarly through either offline binarization or allowing a fixed maximum number of gap words between the right hand side nonterminals in the decoder.

For these reasons, the parsing problems for more realistic synchronous CFGs such as in Chiang (2005) and Galley et al. (2006) are formally equivalent to ITG. Therefore, we believe our focus on ITG for the search efficiency issue is likely to generalize to other SCFG-based methods.

Without an n-gram language model, decoding using SCFG is not much different from CFG parsing. At each time a CFG rule is applied on the input string, we apply the synchronized CFG rule for the output language. From a dynamic programming point of view, the DP states are $X[i,j]$, where X ranges over all possible nonterminals and i and j range over 0 to the input string length $|w|$. Each state stores the best translations obtainable. When we reach the top state $S[0,|w|]$, we can get the best translation for the entire sentence. The algorithm is $O(|w|^3)$.

However, when we want to integrate an n-gram language model into the search, our goal is searching for the derivation whose total sum of weights of productions and n-gram log probabilities is maximized. Now the adjacent span-parameterized states $X[i,k]$ and $X[k,j]$ can interact with each other by "peeping into" the leading and trailing $n-1$ words on the output side for each state. Different boundary words differentiate the span-parameterized states. Thus, to preserve the dynamic programming property, we need to refine the states by adding the boundary words into the parameterization. The LM-integrated states are represented as $X[i,j,u_{1,..,n-1},v_{1,..,n-1}]$. Since the number of variables involved at each DP step has increased to $3+4(n-1)$, the decoding algorithm is asymptotically $O(|w|^{3+4(n-1)})$. Although it is possible to use the "hook" trick of Huang et al. (2005) to factorize the DP operations to reduce the complexity to $O(|w|^{3+3(n-1)})$, when n is greater than 2, the complexity is still prohibitive.

3 Multi-pass LM-Integrated Decoding

In this section, we describe a multi-pass progressive decoding technique that gradually augments the LM-integrated states from lower orders to higher orders. For instance, a bigram-integrated state $[X,i,j,u,v]$ is said to be a coarse-level state of a trigram-integrate state $[X,i,j,u,u',v',v]$, because the latter state refines the previous by specifying more inner words.

Progressive search has been used for HMM's in speech recognition (Murveit et al., 1993). The gen-

eral idea is to use a simple and fast decoding algorithm to constrain the search space of a following more complex and slower technique. More specifically, a bigram decoding pass is executed forward and backward to figure out the probability of each state. Then the states can be pruned based on their global score using the product of inside and outside probabilities. The advanced decoding algorithm will use the constrained space (a lattice in the case of speech recognition) as a grammatical constraint to help it focus on a smaller search space on which more discriminative features are brought in.

The same idea has been applied to forests for parsing. Charniak and Johnson (2005) use a PCFG to do a pass of inside-outside parsing to reduce the state space of a subsequent lexicalized n-best parsing algorithm to produce parses that are further re-ranked by a MaxEnt model.

We take the same view as in speech recognition that a trigram integrated model is a finer-grained model than bigram model and in general we can do an $n-1$-gram decoding as a predicative pass for the following n-gram pass. We need to do inside-outside parsing as coarse-to-fine parsers do. However, we use the outside probability or cost information differently. We do not combine the inside and outside costs of a simpler model to prune the space for a more complex model. Instead, for a given finer-gained state, we combine its true inside cost with the outside cost of its coarse-level counter-part to estimate its worthiness of being explored. The use of the outside cost from a coarser-level as the outside estimate makes our method naturally fall in the framework of A* parsing.

Klein and Manning (2003) describe an A* parsing framework for monolingual parsing and admissible outside estimates that are computed using inside/outside parsing algorithm on simplified PCFGs compared to the original PCFG. Zhang and Gildea (2006) describe A* for ITG and develop admissible heuristics for both alignment and decoding. Both have shown the effectiveness of A* in situations where the outside estimate approximates the true cost closely such as when the sentences are short. For decoding long sentences, it is difficult to come up with good admissible (or inadmissible) heuristics. If we can afford a bigram decoding pass, the outside cost from a bigram model is conceivably a very good estimate of the outside cost using a trigram model since a bigram language model and a trigram language model must be strongly correlated. Although we lose the guarantee that the bigram-pass outside estimate is admissible, we expect that it approximates the outside cost very closely, thus very likely to effectively guide the heuristic search.

3.1 Inside-outside Coarse Level Decoding

We describe the coarse level decoding pass in this section. The decoding algorithms for the coarse level and the fine level do not necessarily have to be the same. The fine level decoding algorithm is an A* algorithm. The coarse level decoding algorithm can be CKY or A* or other alternatives.

Conceptually, the algorithm is finding the shortest hyperpath in the hypergraph in which the nodes are states like $X[i, j, u_{1,..,n-1}, v_{1,..,n-1}]$, and the hyperedges are the applications of the synchronous rules to go from right-hand side states to left-hand side states. The root of the hypergraph is a special node $S'[0, |w|, \langle s\rangle, \langle /s\rangle]$ which means the entire input sentence has been translated to a string starting with the beginning-of-sentence symbol and ending at the end-of-sentence symbol. If we imagine a starting node that goes to all possible basic translation pairs, i.e., the instances of the terminal translation rules for the input, we are searching the shortest hyper path from the imaginary bottom node to the root. To help our outside parsing pass, we store the back-pointers at each step of exploration.

The outside parsing pass, however, starts from the root $S'[|w|, \langle s\rangle, \langle /s\rangle]$ and follows the back-pointers downward to the bottom nodes. The nodes need to be visited in a topological order so that whenever a node is visited, its parents have been visited and its outside cost is over all possible outside parses. The algorithm is described in pseudocode in Algorithm 1. The number of hyperedges to traverse is much fewer than in the inside pass because not every state explored in the bottom up inside pass can finally reach the goal. As for normal outside parsing, the operations are the reverse of inside parsing. We propagate the outside cost of the parent to its children by combining with the inside cost of the other children and the interaction cost, i.e., the language model cost between the focused child and the other children. Since we want to approximate the Viterbi

outside cost, it makes sense to maximize over all possible outside costs for a given node, to be consistent with the maximization of the inside pass. For the nodes that have been explored in the bottom up pass but not in the top-down pass, we set their outside cost to be infinity so that their exploration is preferred only when the viable nodes from the first pass have all been explored in the fine pass.

3.2 Heuristics for Fine-grained Decoding

In this section, we summarize the heuristics for finer level decoding.

The motivation for combining the true inside cost of the fine-grained model and the outside estimate given by the coarse-level parsing is to approximate the true global cost of a fine-grained state as closely as possible. We can make the approximation even closer by incorporating local higher-order outside n-gram information for a state of $X[i, j, u_{1,..,n-1}, v_{1,..,n-1}]$ into account. We call this the *best-border* estimate. For example, the best-border estimate for trigram states is:

$$\begin{aligned} & h_{BB}(X, i, j, u_1, u_2, v_1, v_2) \\ = & \left[\max_{s \in S(i,j)} P_{lm}(u_2 \mid s, u_1)\right] \\ & \cdot \left[\max_{s \in S(i,j)} P_{lm}(s \mid v_1, v_2)\right] \end{aligned}$$

where $S(i, j)$ is the set of candidate target language words outside the span of (i, j). h_{BB} is the product of the upper bounds for the two on-the-border n-grams.

This heuristic function was one of the admissible heuristics used by Zhang and Gildea (2006). The benefit of including the best-border estimate is to refine the outside estimate with respect to the inner words which refine the bigram states into the trigram states. If we do not take the inner words into consideration when computing the outside cost, all states that map to the same coarse level state would have the same outside cost. When the simple best-border estimate is combined with the coarse-level outside estimate, it can further boost the search as will be shown in the experiments. To summarize, our recipe for faster decoding is that using

$$\begin{aligned} & \beta(X[i, j, u_{1,..,n-1}, v_{1,..,n-1}]) \\ & \quad + \quad \alpha(X[i, j, u_1, v_{n-1}]) \\ & \quad + \quad h_{BB}(X, i, j, u_{1,...,n}, v_{1,...,n}) \end{aligned} \quad (1)$$

where β is the Viterbi inside cost and α is the Viterbi outside cost, to globally prioritize the n-gram integrated states on the agenda for exploration.

3.3 Alternative Efficient Decoding Algorithms

The complexity of n-gram integrated decoding for SCFG has been tackled using other methods.

The hook trick of Huang et al. (2005) factorizes the dynamic programming steps and lowers the asymptotic complexity of the n-gram integrated decoding, but has not been implemented in large-scale systems where massive pruning is present.

The cube-pruning by Chiang (2007) and the lazy cube-pruning of Huang and Chiang (2007) turn the computation of beam pruning of CYK decoders into a top-k selection problem given two columns of translation hypotheses that need to be combined. The insight for doing the expansion top-down lazily is that there is no need to uniformly explore every cell. The algorithm starts with requesting the first best hypothesis from the root. The request translates into requests for the k-bests of some of its children and grandchildren and so on, because re-ranking at each node is needed to get the top ones.

Venugopal et al. (2007) also take a two-pass decoding approach, with the first pass leaving the language model boundary words out of the dynamic programming state, such that only one hypothesis is retained for each span and grammar symbol.

4 Decoding to Maximize BLEU

The ultimate goal of efficient decoding to find the translation that has a highest evaluation score using the least time possible. Section 3 talks about utilizing the outside cost of a lower-order model to estimate the outside cost of a higher-order model, boosting the search for the higher-order model. By doing so, we hope the intrinsic metric of our model agrees with the extrinsic metric of evaluation so that fast search for the model is equivalent to efficient decoding. But the mismatch between the two is evident, as we will see in the experiments. In this section,

Algorithm 1 OutsideCoarseParsing()

for all $X[i,j,u,v]$ in topological order **do**
  for all children pairs pointed to by the back-pointers **do**
    if $X \rightarrow [YZ]$ **then**
      $\triangleright$ the two children are $Y[i,k,u,u']$ and $Z[k,j,v',v]$
      $\alpha(Y[i,k,u,u']) = \max\{\alpha(Y[i,k,u,u']),$
        $\alpha(X[i,j,u,v]) + \beta(Z[k,j,v',v]) + rule(X \rightarrow [YZ]) + bigram(u',v')\}$
      $\alpha(Z[k,j,v',v]) = \max\{\alpha(Z[k,j,v',v]),$
        $\alpha(X[i,j,u,v]) + \beta(Y[i,k,u,u']) + rule(X \rightarrow [YZ]) + bigram(u',v')\}$
    end if
    if $X \rightarrow \langle YZ \rangle$ **then**
      $\triangleright$ the two children are $Y[i,k,v',v]$ and $Z[k,j,u,u']$
      $\alpha(Y[i,k,v',v]) = \max\{\alpha(Y[i,k,v',v]),$
        $\alpha(X[i,j,u,v]) + \beta(Z[k,j,u,u']) + rule(X \rightarrow \langle YZ \rangle) + bigram(u',v')\}$
      $\alpha(Z[k,j,u,u']) = \max\{\alpha(Z[k,j,u,u']),$
        $\alpha(X[i,j,u,v]) + \beta(Y[i,k,v',v]) + rule(X \rightarrow \langle YZ \rangle) + bigram(u',v')\}$
    end if
  end for
end for

we deal with the mismatch by introducing another decoding pass that maximizes the expected count of synchronous constituents in the tree corresponding to the translation returned. BLEU is based on n-gram precision, and since each synchronous constituent in the tree adds a new 4-gram to the translation at the point where its children are concatenated, the additional pass approximately maximizes BLEU.

Kumar and Byrne (2004) proposed the framework of Minimum Bayesian Risk (MBR) decoding that minimizes the expected loss given a loss function. Their MBR decoding is a reranking pass over an n-best list of translations returned by the decoder. Our algorithm is another dynamic programming decoding pass on the trigram forest, and is similar to the parsing algorithm for maximizing expected labelled recall presented by Goodman (1996).

4.1 Maximizing the expected count of correct synchronous constituents

We introduce an algorithm that maximizes the expected count of correct synchronous constituents. Given a synchronous constituent specified by the state $[X,i,j,u,u',v',v]$, its probability of being correct in the model is

$$EC([X,i,j,u,u',v',v]) = \alpha([X,i,j,u,u',v',v]) \cdot \beta([X,i,j,u,u',v',v]),$$

where α is the outside probability and β is the inside probability. We approximate β and α using the Viterbi probabilities. Since decoding from bottom up in the trigram pass already gives us the inside Viterbi scores, we only have to visit the nodes in the reverse order once we reach the root to compute the Viterbi outside scores. The outside-pass Algorithm 1 for bigram decoding can be generalized to the trigram case. We want to maximize over all translations (synchronous trees) T in the forest after the trigram decoding pass according to

$$\max_{T} \sum_{[X,i,j,u,u',v',v] \in T} EC([X,i,j,u,u',v',v]).$$

The expression can be factorized and computed using dynamic programming on the forest.

5 Experiments

We did our decoding experiments on the LDC 2002 MT evaluation data set for translation of Chinese newswire sentences into English. The evaluation data set has 10 human translation references for each sentence. There are a total of 371 Chinese sentences of no more than 20 words in the data set. These sentences are the test set for our different versions of language-model-integrated ITG decoders. We evaluate the translation results by comparing them against the reference translations using the BLEU metric.

The word-to-word translation probabilities are from the translation model of IBM Model 4 trained on a 160-million-word English-Chinese parallel corpus using GIZA++. The phrase-to-phrase translation probabilities are trained on 833K parallel sentences. 758K of this was data made available by ISI, and another 75K was FBIS data. The language model is trained on a 30-million-word English corpus. The rule probabilities for ITG are trained using EM on a corpus of 18,773 sentence pairs with a total of 276,113 Chinese words and 315,415 English words.

5.1 Bigram-pass Outside Cost as Trigram-pass Outside Estimate

We first fix the beam for the bigram pass, and change the outside heuristics for the trigram pass to show the difference before and after using the first-pass outside cost estimate and the border estimate. We choose the beam size for the CYK bigram pass to be 10 on the log scale. The first row of Table 1 shows the number of explored hyperedges for the bigram pass and its BLEU score. In the rows below, we compare the additional numbers of hyperedges that need to be explored in the trigram pass using different outside heuristics. It takes too long to finish using uniform outside estimate; we have to use a tight beam to control the agenda-based exploration. Using the bigram outside cost estimate makes a huge difference. Furthermore, using Equation 1, adding the additional heuristics on the best trigrams that can appear on the borders of the current hypothesis, on average we only need to explore 2700 additional hyperedges per sentence to boost the BLEU score from 21.77 to 23.46. The boost is so significant that overall the dominant part of search time is no longer the second pass but the first bigram pass (inside pass actually) which provides a constrained space and outside heuristics for the second pass.

5.2 Two-pass decoding versus One-pass decoding

By varying the beam size for the first pass, we can plot graphs of model scores versus search time and BLEU scores versus search time as shown in Figure 1. We use a very large beam for the second pass due to the reason that the outside estimate for the second pass is discriminative enough to guide the search. We sum up the total number of seconds for both passes to compare with the baseline systems. On average, less than 5% of time is spent in the second pass.

Decoding Method	Avg. Hyperedges	BLEU
Bigram Pass	167K	21.77
Trigram Pass		
UNI	–	–
BO	+ 629.7K=796.7K	23.56
BO+BB	*+2.7K =169.7K*	*23.46*
Trigram One-pass, with Beam	6401K	23.47

Table 1: Speed and BLEU scores for two-pass decoding. UNI stands for the uniform (zero) outside estimate. BO stands for the bigram outside cost estimate. BB stands for the best border estimate, which is added to BO.

Decoder	Time	BLEU	Model Score
One-pass agenda	4317s	22.25	-208.849
One-pass CYK	3793s	22.89	-207.309
Multi-pass, CYK first agenda second pass	*3689s*	*23.56*	*-205.344*
MEC third pass	**3749s**	**24.07**	**-203.878**
Lazy-cube-pruning	3746s	22.16	-208.575

Table 2: Summary of different trigram decoding strategies, using about the same time (10 seconds per sentence).

In Figure 1, we have four competing decoders. *bitri_cyk* is our two-pass decoder, using CYK as the first pass decoding algorithm and using agenda-based decoding in the second pass which is guided by the first pass. *agenda* is our trigram-integrated agenda-based decoder. The other two systems are also one-pass. *cyk* is our trigram-integrated CYK decoder. *lazy kbest* is our top-down k-best-style decoder.[1]

Figure 1(left) compares the search efficiencies of the four systems. *bitri_cyk* at the top ranks first. *cyk* follows it. The curves of *lazy kbest* and *agenda* cross

[1]In our implementation of the lazy-cube-pruning based ITG decoder, we vary the re-ranking buffer size and the the top-k list size which are the two controlling parameters for the search space. But we did not use any LM estimate to achieve early stopping as suggested by Huang and Chiang (2007). Also, we did not have a translation-model-only pruning pass. So the results shown in this paper for the lazy cube pruning method is not of its best performance.

and are both below the curves of *bitri_cyk* and *cyk*. This figure indicates the advantage of the two-pass decoding strategy in producing translations with a high model score in less time.

However, model scores do not directly translate into BLEU scores. In Figure 1(right), *bitri_cyk* is better than *CYK* only in a certain time window when the beam is neither too small nor too large. But the window is actually where we are interested – it ranges from 5 seconds per sentence to 20 seconds per sentence. Table 2 summarizes the performance of the four decoders when the decoding speed is at 10 seconds per sentence.

5.3 Does the hook trick help?

We have many choices in implementing the bigram decoding pass. We can do either CYK or agenda-based decoding. We can also use the dynamic programming hook trick. We are particularly interested in the effect of the hook trick in a large-scale system with aggressive pruning.

Figure 2 compares the four possible combinations of the decoding choices for the first pass: *bitri_cyk*, *bitri_agenda*, *bitri_cyk_hook* and *bitri_agenda_hook*. *bitri_cyk* which simply uses CYK as the first pass decoding algorithm is the best in terms of performance and time trade-off. The hook-based decoders do not show an advantage in our experiments. Only *bitri_agenda_hook* gets slightly better than *bitri_agenda* when the beam size increases. So, it is very likely the overhead of building hooks offsets its benefit when we massively prune the hypotheses.

5.4 Maximizing BLEU

The *bitri_cyk* decoder spends little time in the agenda-based trigram pass, quickly reaching the goal item starting from the bottom of the chart. In order to maximize BLEU score using the algorithm described in Section 4, we need a sizable trigram forest as a starting point. Therefore, we keep popping off more items from the agenda after the goal is reached. Simply by exploring more (200 times the log beam) after-goal items, we can optimize the Viterbi synchronous parse significantly, shown in Figure 3(left) in terms of model score versus search time.

However, the mismatch between model score and BLEU score persists. So, we try our algorithm of maximizing expected count of synchronous constituents on the trigram forest. We find significant improvement in BLEU, as shown in Figure 3 (right) by the curve of *bitri_cyk_epass_me_cons*. *bitri_cyk_epass_me_cons* beats both *bitri_cyk* and *cyk* in terms of BLEU versus time if using more than 1.5 seconds on average to decode each sentence. At each time point, the difference in BLEU between *bitri_cyk_epass_me_cons* and the highest of *bitri_cyk* and *cyk* is around .5 points consistently as we vary the beam size for the first pass. We achieve the record-high BLEU score 24.34 using on average 21 seconds per sentence, compared to the next-highest score of 23.92 achieved by *cyk* using on average 78 seconds per sentence.

6 Conclusion

We present a multi-pass method to speed up n-gram integrated decoding for SCFG. We use an inside/outside parsing algorithm to get the Viterbi outside cost of bigram integrated states which is used as an outside estimate for trigram integrated states. The coarse-level outside cost plus the simple estimate for border trigrams speeds up the trigram decoding pass hundreds of times compared to using no outside estimate.

Maximizing the probability of the synchronous derivation is not equivalent to maximizing BLEU. We use a rescoring decoding pass that maximizes the expected count of synchronous constituents. This technique, together with the progressive search at previous stages, gives a decoder that produces the highest BLEU score we have obtained on the data in a very reasonable amount of time.

As future work, new metrics for the final pass may be able to better approximate BLEU. As the bigram decoding pass currently takes the bulk of the decoding time, better heuristics for this phase may speed up the system further.

Acknowledgments This work was supported by NSF ITR-0428020 and NSF IIS-0546554.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*.

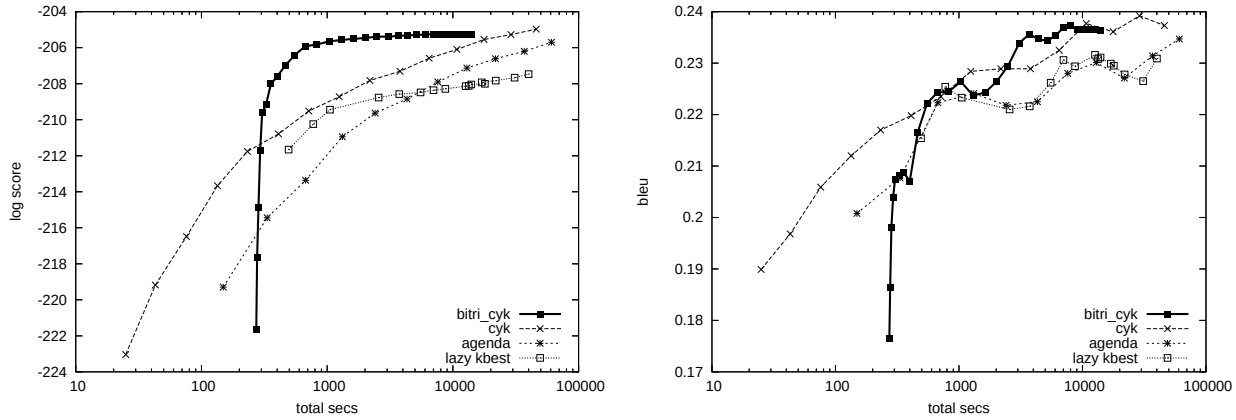

Figure 1: We compare the two-pass ITG decoder with the one-pass trigram-integrated ITG decoders in terms of both model scores vs. time (left) and BLEU scores vs. time (right). The model score here is the log probability of the decoded parse, summing up both the translation model and the language model. We vary the beam size (for the first pass in the case of two-pass) to search more and more thoroughly.

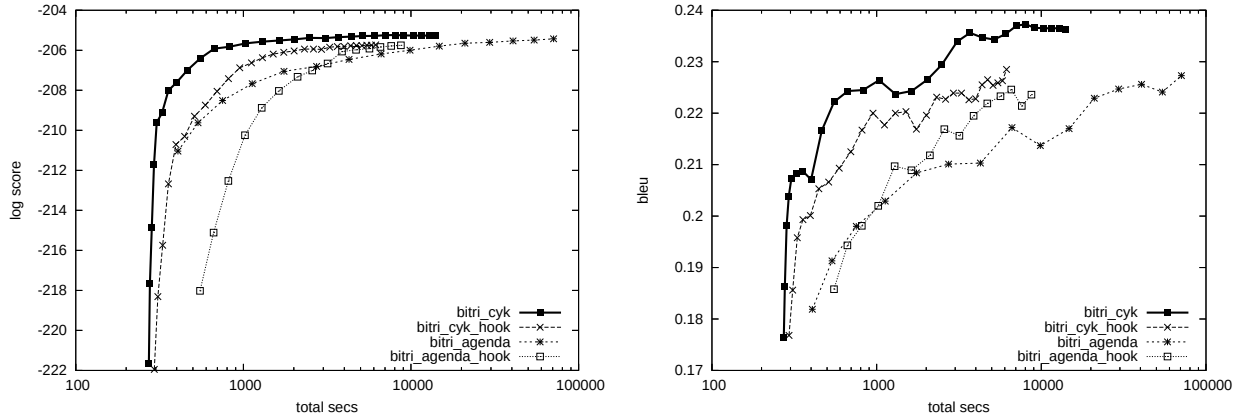

Figure 2: We use different first-pass decoding algorithms, fixing the second pass to be agenda-based which is guided by the outside cost of the first pass. Left: model score vs. time. Right: BLEU score vs. time.

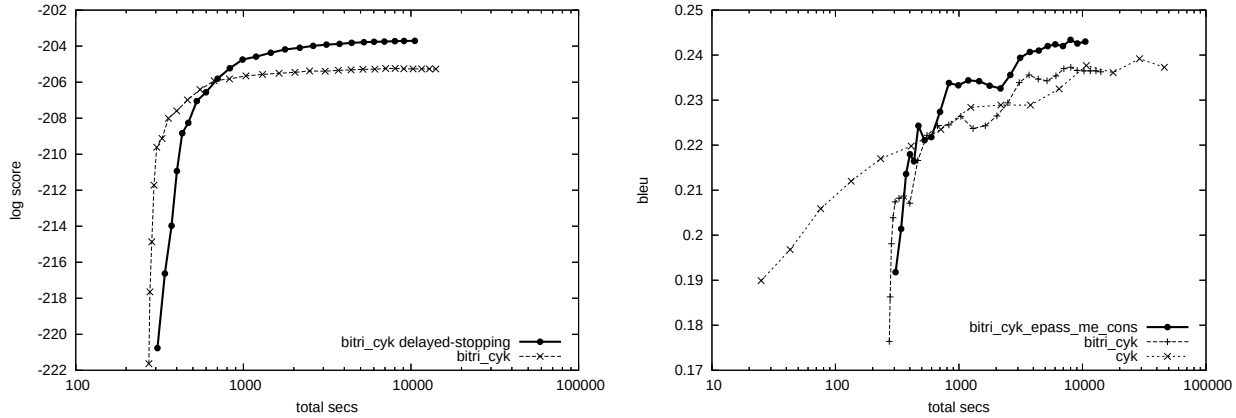

Figure 3: Left: improving the model score by extended agenda-exploration after the goal is reached in the best-first search. Right: maximizing BLEU by the maximizing expectation pass on the expanded forest.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, pages 263–270.

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

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the International Conference on Computational Linguistics/Association for Computational Linguistics (COLING/ACL-06)*, pages 961–968, July.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the 34th Annual Conference of the Association for Computational Linguistics (ACL-96)*, pages 177–183.

Liang Huang and David Chiang. 2007. Faster algorithms for decoding with integrated language models. In *Proceedings of ACL*, Prague, June.

Liang Huang, Hao Zhang, and Daniel Gildea. 2005. Machine translation as lexicalized parsing with hooks. In *International Workshop on Parsing Technologies (IWPT05)*, Vancouver, BC.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*.

Shankar Kumar and William Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 169–176, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Hy Murveit, John W. Butzberger, Vassilios V. Digalakis, and Mitchel Weintraub. 1993. Large-vocabulary dictation using SRI's decipher speech recognition system: Progressive-search techniques. In *Proceedings of the IEEE International Conference on Acoustics, Speech, & Signal Processing (IEEE ICASSP-93)*, volume 2, pages 319–322. IEEE.

Giorgio Satta and Enoch Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 803–810, Vancouver, Canada, October.

Ashish Venugopal, Andreas Zollmann, and Stephan Vogel. 2007. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *NAACL07*, Rochester, NY, April.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Hao Zhang and Daniel Gildea. 2006. Efficient search for inversion transduction grammar. In *2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sydney.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the 2006 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-06)*, pages 256–263.