Coarse-to-Fine Syntactic Machine Translation using Language Projections

Slav Petrov **Aria Haghighi** **Dan Klein**
Computer Science Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720
{petrov, aria42, klein}@eecs.berkeley.edu

Abstract

The intersection of tree transducer-based translation models with n-gram language models results in huge dynamic programs for machine translation decoding. We propose a multipass, coarse-to-fine approach in which the language model complexity is incrementally introduced. In contrast to previous *order-based* bigram-to-trigram approaches, we focus on *encoding-based* methods, which use a clustered encoding of the target language. Across various encoding schemes, and for multiple language pairs, we show speed-ups of up to 50 times over single-pass decoding while improving BLEU score. Moreover, our entire decoding cascade for trigram language models is faster than the corresponding bigram pass alone of a bigram-to-trigram decoder.

1 Introduction

In the absence of an n-gram language model, decoding a synchronous CFG translation model is very efficient, requiring only a variant of the CKY algorithm. As in monolingual parsing, dynamic programming items are simply indexed by a source language span and a syntactic label. Complexity arises when n-gram language model scoring is added, because items must now be distinguished by their initial and final few target language words for purposes of later combination. This lexically exploded search space is a root cause of inefficiency in decoding, and several methods have been suggested to combat it. The approach most relevant to the current work is Zhang and Gildea (2008), which begins with an initial bigram pass and uses the resulting chart to guide a final trigram pass. Substantial speed-ups are obtained, but computation is still dominated by the initial bigram pass. The key challenge is that unigram models are too poor to prune well, but bigram models are already huge. In short, the problem is that there are too many words in the target language. In this paper, we propose a new, coarse-to-fine, multipass approach which allows much greater speed-ups by translating into *abstracted languages*. That is, rather than beginning with a *low-order* model of a still-large language, we exploit *language projections*, hierarchical clusterings of the target language, to effectively reduce the size of the target language. In this way, initial passes can be very quick, with complexity phased in gradually.

Central to coarse-to-fine language projection is the construction of sequences of word clusterings (see Figure 1). The clusterings are deterministic mappings from words to clusters, with the property that each clustering refines the previous one. There are many choice points in this process, including how these clusterings are obtained and how much refinement is optimal for each pass. We demonstrate that likelihood-based hierarchical EM training (Petrov et al., 2006) and cluster-based language modeling methods (Goodman, 2001) are superior to both rank-based and random-projection methods. In addition, we demonstrate that more than two passes are beneficial and show that our computation is equally distributed over all passes. In our experiments, passes with less than 16-cluster language models are most advantageous, and even a single pass with just two word clusters can reduce decoding time greatly.

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 108–116,
Honolulu, October 2008.

To follow related work and to focus on the effects of the language model, we present translation results under an inversion transduction grammar (ITG) translation model (Wu, 1997) trained on the Europarl corpus (Koehn, 2005), described in detail in Section 3, and using a trigram language model. We show that, on a range of languages, our coarse-to-fine decoding approach greatly outperforms baseline beam pruning and bigram-to-trigram pruning on time-to-BLEU plots, reducing decoding times by up to a factor of 50 compared to single pass decoding. In addition, coarse-to-fine decoding increases BLEU scores by up to 0.4 points. This increase is a mixture of improved search and subtly advantageous coarse-to-fine effects which are further discussed below.

2 Coarse-to-Fine Decoding

In coarse-to-fine decoding, we create a series of initially simple but increasingly complex search problems. We then use the solutions of the simpler problems to prune the search spaces for more complex models, reducing the total computational cost.

2.1 Related Work

Taken broadly, the coarse-to-fine approach is not new to machine translation (MT) or even syntactic MT. Many common decoder precomputations can be seen as coarse-to-fine methods, including the A*-like forward estimates used in the Moses decoder (Koehn et al., 2007). In an ITG framework like ours, Zhang and Gildea (2008) consider an approach in which the results of a bigram pass are used as an A* heuristic to guide a trigram pass. In their two-pass approach, the coarse bigram pass becomes computationally dominant. Our work differs in two ways. First, we use posterior pruning rather than A* search. Unlike A* search, posterior pruning allows multipass methods. Not only are posterior pruning methods simpler (for example, there is no need to have complex multipart bounds), but they can be much more effective. For example, in monolingual parsing, posterior pruning methods (Goodman, 1997; Charniak et al., 2006; Petrov and Klein, 2007) have led to greater speedups than their more cautious A* analogues (Klein and Manning, 2003; Haghighi et al., 2007), though at the cost of guaranteed optimality.

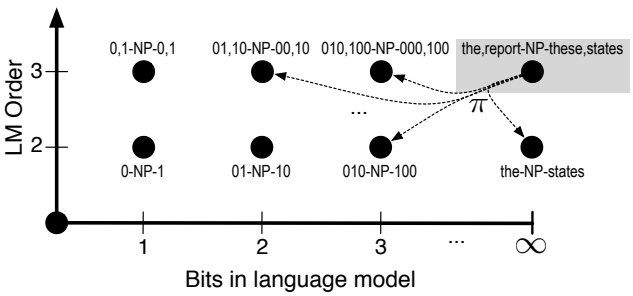

Figure 2: Possible state projections π for the target noun phrase *"the report for these states"* using the clusters from Figure 1. The number of bits used to encode the target language vocabulary is varied along the x-axis. The language model order is varied along the y-axis.

Second, we focus on an orthogonal axis of abstraction: the size of the target language. The introduction of abstract languages gives better control over the granularity of the search space and provides a richer set of intermediate problems, allowing us to adapt the level of refinement of the intermediate, coarse passes to minimize total computation.

Beyond coarse-to-fine approaches, other related approaches have also been demonstrated for syntactic MT. For example, Venugopal et al. (2007) considers a greedy first pass with a full model followed by a second pass which bounds search to a region near the greedy results. Huang and Chiang (2007) searches with the full model, but makes assumptions about the the amount of reordering the language model can trigger in order to limit exploration.

2.2 Language Model Projections

When decoding in a syntactic translation model with an n-gram language model, search states are specified by a grammar nonterminal X as well as the the n-1 left-most target side words $l_{n-1}, \ldots, l_1$ and right-most target side words $r_1, \ldots, r_{n-1}$ of the generated hypothesis. We denote the resulting lexicalized state as $l_{n-1}, \ldots, l_1$-X-$r_1, \ldots, r_{n-1}$. Assuming a vocabulary V and grammar symbol set G, the state space size is up to $|V|^{2(n-1)}|G|$, which is immense for a large vocabulary when $n > 1$. We consider two ways to reduce the size of this search space. First, we can reduce the order of the language model. Second, we can reduce the number of words in the vocabulary. Both can be thought of as *projections* of the search space to smaller ab-

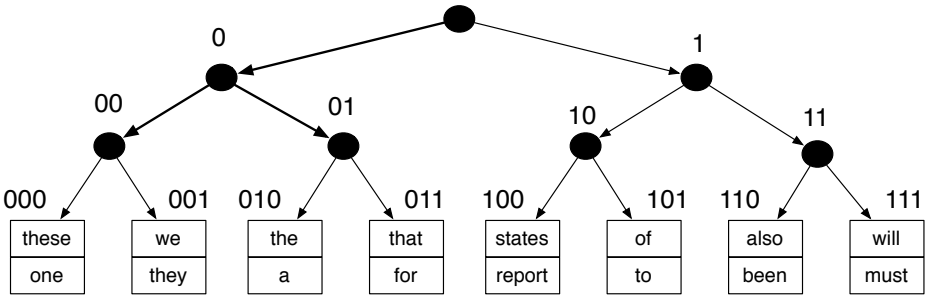

Figure 1: An example of hierarchical clustering of target language vocabulary (see Section 4). Even with a small number of clusters our divisive HMM clustering (Section 4.3) captures sensible syntactico-semantic classes.

stracted spaces. Figure 2 illustrates those two orthogonal axes of abstraction.

Order-based projections are simple. As shown in Figure 2, they simply strip off the appropriate words from each state, collapsing dynamic programming items which are identical from the standpoint of their left-to-right combination in the lower order language model. However, having only order-based projections is very limiting. Zhang and Gildea (2008) found that their computation was dominated by their bigram pass. The only lower-order pass possible uses a unigram model, which provides no information about the interaction of the language model and translation model reorderings. We therefore propose *encoding-based* projections. These projections reduce the size of the target language vocabulary by deterministically projecting each target language word to a word cluster. This projection extends to the whole search state in the obvious way: assuming a bigram language model, the state l-X-r projects to $c(l)$-X-$c(r)$, where $c(\cdot)$ is the deterministic word-to-cluster mapping.

In our multipass approach, we will want a sequence $c_1 \ldots c_n$ of such projections. This requires a *hierarchical clustering* of the target words, as shown in Figure 1. Each word's cluster membership can be represented by an n-bit binary string. Each prefix of length k declares that word's cluster assignment at the k-bit level. As we vary k, we obtain a sequence of projections $c_k(\cdot)$, each one mapping words to a more refined clustering. When performing inference in a k-bit projection, we replace the detailed original language model over words with a coarse language model LM_k over the k-bit word clusters. In addition, we replace the phrase table with a projected phrase table, which further increases the speed of projected passes. In Section 4, we describe the various clustering schemes explored, as well as how the coarse LM_k are estimated.

2.3 Multipass Decoding

Unlike previous work, where the state space exists only at two levels of abstraction (i.e. bigram and trigram), we have multiple levels to choose from (Figure 2). Because we use both encoding-based and order-based projections, our options form a lattice of coarser state spaces, varying from extremely simple (a bigram model with just two word clusters) to nearly the full space (a trigram model with 10 bits or 1024 word clusters).

We use this lattice to perform a series of coarse passes with increasing complexity. More formally, we decode a source sentence multiple times, in a sequence of state spaces $S_0, S_1, \ldots, S_n$=S, where each S_i is a refinement of S_{i-1} in either language model order, language encoding size, or both. The state spaces S_i and S_j ($i < j$) are related to each other via a projection operator $\pi_{j \to i}(\cdot)$ which maps refined states deterministically to coarser states.

We start by decoding an input x in the simplest state space S_0. In particular, we compute the chart of the posterior distributions $p_0(s) = P(s|x)$ for all states $s \in S_0$. These posteriors will be used to prune the search space S_1 of the following pass. States s whose posterior falls below a threshold t trigger the removal of all more refined states s' in the subsequent pass (see Figure 3). This technique is *posterior pruning*, and is different from A* methods in two main ways. First, it can be iterated in a multipass setting, and, second, it is generally more effi-

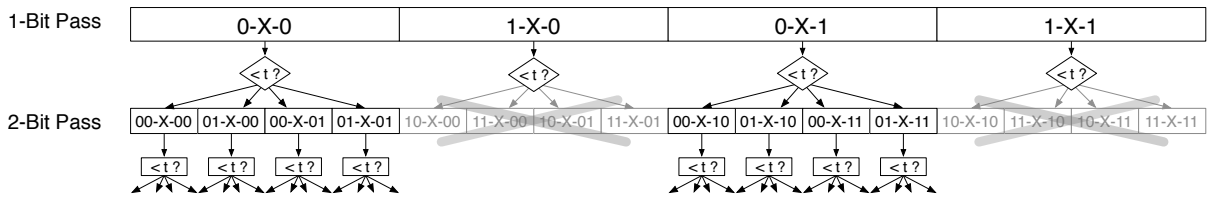

Figure 3: Example of state pruning in coarse-to-fine decoding using the language encoding projection (see Section 2.2). During the coarse one-bit word cluster pass, two of the four possible states are pruned. Every extension of the pruned one-bit states (indicated by the grey shading) are not explored during the two-bit word cluster pass.

cient with a potential cost of increased search errors (see Section 2.1 for more discussion).

Looking at Figure 2, multipass coarse-to-fine decoding can be visualized as a walk from a coarse point somewhere in the lower left to the most refined point in the upper right of the grid. Many coarse-to-fine schedules are possible. In practice, we might start decoding with a 1-bit word bigram pass, followed by an 3-bit word bigram pass, followed by a 5-bit word trigram pass and so on (see Section 5.3 for an empirical investigation). In terms if time, we show that coarse-to-fine gives substantial speed-ups. There is of course an additional memory requirement, but it is negligible. As we will see in our experiments (Section 5) the largest gains can be obtained with extremely coarse language models. In particular, the largest coarse model we use in our best multipass decoder uses a 4-bit encoding and hence has only 16 distinct words (or at most 4096 trigrams).

3 Inversion Transduction Grammars

While our approach applies in principle to a variety of machine translation systems (phrase-based or syntactic), we will use the inversion transduction grammar (ITG) approach of Wu (1997) to facilitate comparison with previous work (Zens and Ney, 2003; Zhang and Gildea, 2008) as well as to focus on language model complexity. ITGs are a subclass of synchronous context-free grammars (SCFGs) where there are only three kinds of rules. Preterminal unary productions produce terminal strings on both sides (words or phrases): $X \rightarrow \mathbf{e}/\mathbf{f}$. Binary in-order productions combine two phrases monotonically ($X \rightarrow [YZ]$). Finally, binary inverted productions invert the order of their children ($X \rightarrow \langle YZ \rangle$). These productions are associated with rewrite weights in the standard way.

Without a language model, SCFG decoding is just like (monolingual) CFG parsing. The dynamic programming states are specified by ${}_iX_j$, where $\langle i, j \rangle$ is a source sentence span and X is a nonterminal. The only difference is that whenever we apply a CFG production on the source side, we need to remember the corresponding synchronous production on the target side and store the best obtainable translation via a backpointer. See Wu (1996) or Melamed (2004) for a detailed exposition.

Once we integrate an n-gram language model, the state space becomes lexicalized and combining dynamic programming items becomes more difficult. Each state is now parametrized by the initial and final $n-1$ words in the target language hypothesis: $l_{n-1}, ..., l_1\text{-}{}_iX_j\text{-}r_1, ..., r_{n-1}$. Whenever we combine two dynamic programming items, we need to score the fluency of their concatentation by incorporating the score of any language model features which cross the target side boundaries of the two concatenated items (Chiang, 2005). Decoding with an integrated language model is computationally expensive for two reasons: (1) the need to keep track of a large number of lexicalized hypotheses for each source span, and (2) the need to frequently query the large language model for each hypothesis combination.

Multipass coarse-to-fine decoding can alleviate both computational issues. We start by decoding in an extremely coarse bigram search space, where there are very few possible translations. We compute standard inside/outside probabilities (iS/oS), as follows. Consider the application of non-inverted binary rule: we combine two items $l_b\text{-}{}_iB_k\text{-}r_b$ and $l_c\text{-}{}_kC_j\text{-}r_c$ spanning $\langle i, k \rangle$ and $\langle k, j \rangle$ respectively to form a larger item $l_b\text{-}{}_iA_j\text{-}r_c$, spanning $\langle i, j \rangle$. The

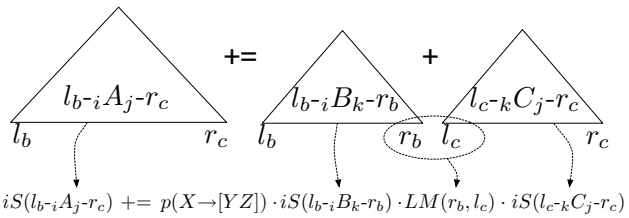

Figure 4: Monotonic combination of two hypotheses during the inside pass involves scoring the fluency of the concatenation with the language model.

inside score of the new item is incremented by:

$$iS(l_b\text{-}_i A_j\text{-}r_c) \mathrel{+}= p(X \to [YZ]) \cdot iS(l_b\text{-}_i B_k\text{-}r_b) \cdot iS(l_c\text{-}_k C_j\text{-}r_c) \cdot LM(r_b, l_c)$$

This process is also illustrated in Figure 4. Of course, we also loop over the split point k and apply the other two rule types (inverted concatenation, terminal generation). We omit those cases from this exposition, as well as the update for the outside pass; they are standard and similar. Once we have computed the inside and outside scores, we compute posterior probabilities for all items:

$$p(l_a\text{-}_i A_j\text{-}r_a) = \frac{iS(l_a\text{-}_i A_j\text{-}r_a) oS(l_a\text{-}_i A_j\text{-}r_a)}{iS(root)}$$

where $iS(root)$ is sum of all translations' scores. States with low posteriors are then pruned away. We proceed to compute inside/outside score in the next, more refined search space, using the projections $\pi_{i \to i-1}$ to map between states in S_i and S_{i-1}. In each pass, we skip all items whose projection into the previous stage had a probability below a stage-specific threshold. This process is illustrated in Figure 3. When we reach the most refined search space S_∞, we do not prune, but rather extract the Viterbi derivation instead.[1]

4 Learning Coarse Languages

Central to our encoding-based projections (see Section 2.2) are hierarchical clusterings of the target language vocabulary. In the present work, these clusterings are each k-bit encodings and yield sequences of coarse language models LM_k and phrasetables PT_k.

[1]Other final decoding strategies are possible, of course, including variational methods and minimum-risk methods (Zhang and Gildea, 2008).

Given a hierarchical clustering, we estimate the corresponding LM_k from a corpus obtained by replacing each token in a target language corpus with the appropriate word cluster. As with our original refined language model, we estimate each coarse language model using the SRILM toolkit (Stolcke, 2002). The phrasetables PT_k are similarly estimated by replacing the words on the target side of each phrase pair with the corresponding cluster. This procedure can potentially map two distinct phrase pairs to the same coarse translation. In such cases we keep only one coarse phrase pair and sum the scores of the colliding originals.

There are many possible schemes for creating hierarchical clusterings. Here, we consider several divisive clustering methods, where coarse word clusters are recursively split into smaller subclusters.

4.1 Random projections

The simplest approach to splitting a cluster is to randomly assign each word type to one of two new subclusters. Random projections have been shown to be a good and computationally inexpensive dimensionality reduction technique, especially for high dimensional data (Bingham and Mannila, 2001). Although our best performance does not come from random projections, we still obtain substantial speed-ups over a single pass fine decoder when using random projections in coarse passes.

4.2 Frequency clustering

In frequency clustering, we allocate words to clusters by frequency. At each level, the most frequent words go into one cluster and the rarest words go into another one. Concretely, we sort the words in a given cluster by frequency and split the cluster so that the two halves have equal token mass. This approach can be seen as a radically simplified version of Brown et al. (1992). It can, and does, result in highly imbalanced cluster hierarchies.

4.3 HMM clustering

An approach found to be effective by Petrov and Klein (2007) for coarse-to-fine parsing is to use likelihood-based hierarchical EM training. We adopt this approach here by identifying each cluster with a latent state in an HMM and determinizing the emissions so that each word type is emitted

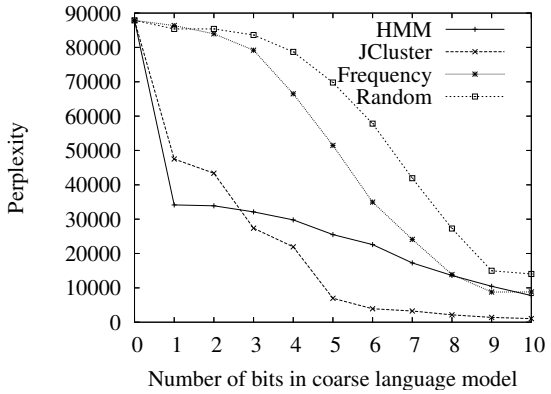

Figure 5: Results of coarse language model perplexity experiment (see Section 4.5). HMM and JClustering have lower perplexity than frequency and random clustering for all number of bits in the language encoding.

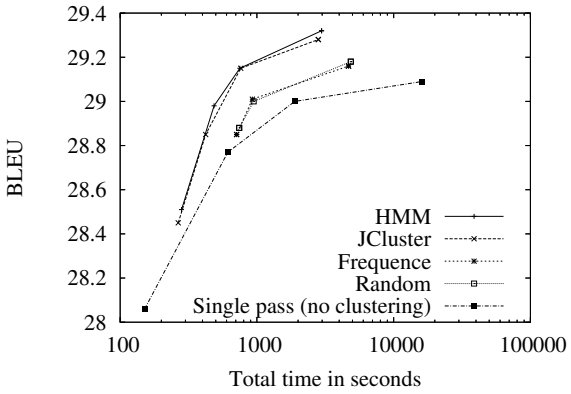

Figure 6: Coarse-to-fine decoding with HMM or JClustering coarse language models reduce decoding times while increasing accuracy.

by only one state. When splitting a cluster s into s_1 and s_2, we initially clone and mildly perturb its corresponding state. We then use EM to learn parameters, which splits the state, and determinize the result. Specifically, each word w is assigned to s_1 if $P(w|s_1) > P(w|s_2)$ and s_2 otherwise. Because of this determinization after each round of EM, a word in one cluster will be allocated to exactly one of that cluster's children. This process not only guarantees that the clusters are hierarchical, it also avoids the state drift discussed by Petrov and Klein (2007). Because the emissions are sparse, learning is very efficient. An example of some of the words associated with early splits can be seen in Figure 1.

4.4 JCluster

Goodman (2001) presents a clustering scheme which aims to minimize the entropy of a word given a cluster. This is accomplished by incrementally swapping words between clusters to locally minimize entropy.[2] This clustering algorithm was developed with a slightly different application in mind, but fits very well into our framework, because the hierarchical clusters it produces are trained to maximize predictive likelihood.

4.5 Clustering Results

We applied the above clustering algorithms to our monolingual language model data to obtain hierarchical clusters. We then trained coarse language models of varying granularity and evaluated them on a held-out set. To measure the quality of the coarse language models we use perplexity (exponentiated cross-entropy).[3] Figure 5 shows that HMM clustering and JClustering have lower perplexity than frequency and random based clustering for all complexities. In the next section we will present a set of machine translation experiments using these coarse language models; the clusterings with better perplexities generally produce better decoders.

5 Experiments

We ran our experiments on the Europarl corpus (Koehn, 2005) and show results on Spanish, French and German to English translation. We used the setup and preprocessing steps detailed in the 2008 Workshop on Statistical Machine Translation.[4] Our baseline decoder uses an ITG with an integrated trigram language model. Phrase translation parameters are learned from parallel corpora with approximately 8.5 million words for each of the language pairs. The English language model is trained on the entire corpus of English parliamentary proceedings provided with the Europarl distribution. We report results on the 2000 development test set sentences of length up to 126 words (average length was 30 words).

[2]The software for this clustering technique is available at `http://research.microsoft.com/~joshuago/`.

[3]We assumed that each cluster had a uniform distribution over all the words in that cluster.

[4]See `http://www.statmt.org/wmt08` for details.

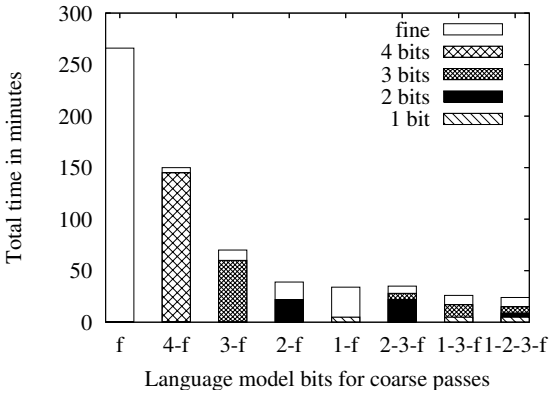

Figure 7: Many passes with extremely simple language models produce the highest speed-ups.

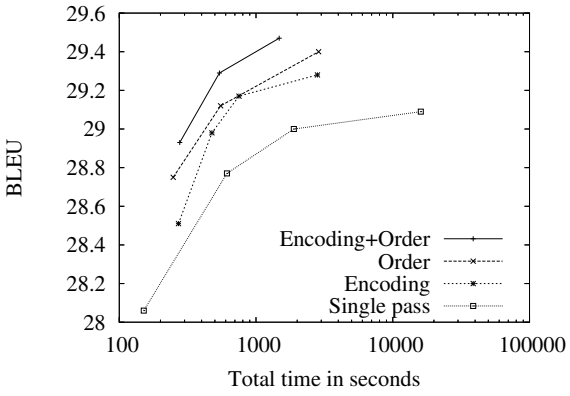

Figure 8: A combination of order-based and encoding-based coarse-to-fine decoding yields the best results.

Our ITG translation model is broadly competitive with state-of-the-art phrase-based-models trained on the same data. For example, on the Europarl development test set, we fall short of Moses (Koehn et al., 2007) by less than one BLEU point. On Spanish-English we get 29.47 BLEU (compared to Moses's 30.40), on French-English 29.34 (vs. 29.95), and 23.80 (vs. 24.64) on German-English. These differences can be attributed primarily to the substantially richer distortion model used by Moses.

The multipass coarse-to-fine architecture that we have introduced presents many choice points. In the following, we investigate various axes individually. We present our findings as BLEU-to-time plots, where the tradeoffs were generated by varying the complexity and the number of coarse passes, as well as the pruning thresholds and beam sizes. Unless otherwise noted, the experiments are on Spanish-English using trigram language models. When different decoder settings are applied to the same model, MERT weights (Och, 2003) from the unprojected single pass setup are used and are kept constant across runs. In particular, the same MERT weights are used for all coarse passes; note that this slightly disadvantages the multipass runs, which use MERT weights optimized for the single pass decoder.

5.1 Clustering

In section Section 4, HMM clustering and JClustering gave lower perplexities than frequency and random clustering when using the same number of bits for encoding the language model. To test how these models perform at pruning, we ran our decoder several times, varying only the clustering source. In each case, we used a 2-bit trigram model as a single coarse pass, followed by a fine output pass. Figure 6 shows that we can obtain significant improvements over the single-pass baseline regardless of the clustering. To no great surprise, HMM clustering and JClustering yield better results, giving a 30-fold speed-up at the same accuracy, or improvements of about 0.3 BLEU when given the same time as the single pass decoder. We discuss this increase in accuracy over the baseline in Section 5.5. Since the performance differences between those two clustering algorithms are negligible, we will use the simpler HMM clustering in all subsequent experiments.

5.2 Spacing

Given a hierarchy of coarse language models, all trigam for the moment, we need to decide on the number of passes and the granularity of the coarse language models used in each pass. Figure 7 shows how decoding time varies for different multipass schemes to achieve the same translation quality. A single coarse pass with a 4-bit language model cuts decoding time almost in half. However, one can further cut decoding time by starting with even coarser language models. In fact, the best results are achieved by decoding in sequence with 1-, 2- and 3-bit language models before running the final fine trigram pass. Interestingly, in this setting, each pass takes about the same amount of time. A similar observation was reported in the parsing literature, where coarse-to-fine inference with multiple passes

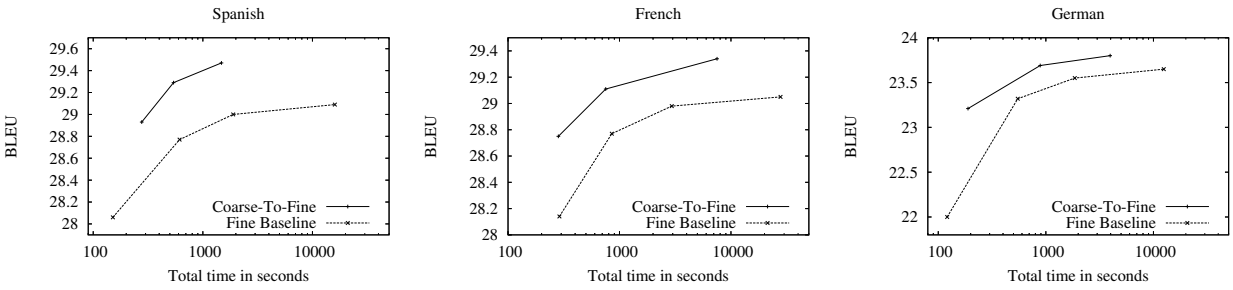

Figure 9: Coarse-to-fine decoding is faster than single pass decoding with a trigram language model and leads to better BLEU scores on all language pairs and for all parameter settings.

of roughly equal complexity produces tremendous speed-ups (Petrov and Klein, 2007).

5.3 Encoding vs. Order

As described in Section 2, the language model complexity can be reduced either by decreasing the vocabulary size (encoding-based projection) or by lowering the language model order from trigram to bigram (order-based projection). Figure 7 shows that both approaches alone yield comparable improvements over the single pass baseline. Fortunately, the two approaches are complimentary, allowing us to obtain further improvements by combining both. We found it best to first do a series of coarse bigram passes, followed by a fine bigram pass, followed by a fine trigram pass.

5.4 Final Results

Figure 9 compares our multipass coarse-to-fine decoder using language refinement to single pass decoding on three different languages. On each language we get significant improvements in terms of efficiency as well as accuracy. Overall, we can achieve up to 50-fold speed-ups at the same accuracy, or alternatively, improvements of 0.4 BLEU points over the best single pass run.

In absolute terms, our decoder translates on average about two Spanish sentences per second at the highest accuracy setting.[5] This compares favorably to the Moses decoder (Koehn et al., 2007), which takes almost three seconds per sentence.

5.5 Search Error Analysis

In multipass coarse-to-fine decoding, we noticed that in addition to computational savings, BLEU scores tend to improve. A first hypothesis is that coarse-to-fine decoding simply improves search quality, where fewer good items fall off the beam compared to a simple fine pass. However, this hypothesis turns out to be incorrect. Table 1 shows the percentage of test sentences for which the BLEU score or log-likelihood changes when we switch from single pass decoding to coarse-to-fine multipass decoding. Only about 30% of the sentences get translated in the same way (if much faster) with coarse-to-fine decoding. For the rest, coarse-to-fine decoding mostly finds translations with lower likelihood, but higher BLEU score, than single pass decoding.[6] An increase of the underlying objectives of interest when pruning despite an increase in model-score search errors has also been observed in monolingual coarse-to-fine syntactic parsing (Charniak et al., 1998; Petrov and Klein, 2007). This effect may be because coarse-to-fine approximates certain minimum Bayes risk objective. It may also be an effect of model intersection between the various passes' models. In any case, both possibilities are often perfectly desirable. It is also worth noting that the number of search errors incurred in the coarse-to-fine approach can be dramatically reduced (at the cost of decoding time) by increasing the pruning thresholds. However, the fortuitous nature of coarse-to-fine search errors seems to be a substantial and desirable effect.

[5] Of course, the time for an average sentence is much lower, since long sentences dominate the overall translation time.

[6] We compared the influence of multipass decoding on the TM score and the LM score; both decrease.

		LL		
		$>$	$=$	$<$
BLEU	$>$	3.6%	-	26.3%
	$=$	1.5%	29.6 %	12.9 %
	$<$	2.2%	-	24.1%

Table 1: Percentage of sentences for which the BLEU score/log-likelihood improves/drops during coarse-to-fine decoding (compared to single pass decoding).

6 Conclusions

We have presented a coarse-to-fine syntactic decoder which utilizes a novel encoding-based language projection in conjunction with order-based projections to achieve substantial speed-ups. Unlike A* methods, a posterior pruning approach allows multiple passes, which we found to be very beneficial for total decoding time. When aggressively pruned, coarse-to-fine decoding can incur additional search errors, but we found those errors to be fortuitous more often than harmful. Our framework applies equally well to other translation systems, though of course interesting new challenges arise when, for example, the underlying SCFGs become more complex.

References

E. Bingham and H.i Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01*.

P. Brown, V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*.

E. Charniak, S. Goldwater, and M. Johnson. 1998. Edge-based best-first chart parsing. 6^{th} *Workshop on Very Large Corpora*.

E. Charniak, M. Johnson, D. McClosky, et al. 2006. Multi-level coarse-to-fine PCFG Parsing. In *HLT-NAACL '06*.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *ACL '05*.

J. Goodman. 1997. Global thresholding and multiple-pass parsing. In *EMNLP '97*.

J. Goodman. 2001. A bit of progress in language modeling. Technical report, Microsoft Research.

A. Haghighi, J. DeNero, and D. Klein. 2007. A* search via approximate factoring. In *NAACL '07*.

L. Huang and D. Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *ACL '07*.

D. Klein and C. Manning. 2003. A* parsing: fast exact viterbi parse selection. In *NAACL '03*.

P. Koehn, H. Hoang, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL '07*.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

I. D. Melamed. 2004. Statistical machine translation by parsing. In *ACL '04*.

F. Och. 2003. Minimum error rate training in statistical machine translation. In *ACL '03*.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL '07*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL '06*.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *ICSLP '02*.

A. Venugopal, A. Zollmann, and S. Vogel. 2007. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *HLT-NAACL '07*.

D. Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *ACL '96*.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. In *Computational Linguistics*.

R. Zens and H. Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *ACL '03*.

H. Zhang and D. Gildea. 2008. Efficient multi-pass decoding for synchronous context free grammars. In *ACL '08*.