Parsers as language models for statistical machine translation

Matt Post and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

Most work in syntax-based machine translation has been in translation modeling, but there are many reasons why we may instead want to focus on the language model. We experiment with parsers as language models for machine translation in a simple translation model. This approach demands much more of the language models, allowing us to isolate their strengths and weaknesses. We find that unmodified parsers do not improve BLEU scores over ngram language models, and provide an analysis of their strengths and weaknesses.

1 Introduction

In the past few years there has been a burgeoning interest in syntax-based approaches to statistical machine translation. Most of this effort has been directed at the translation model rather than the language model; however, the level of syntactic divergence in parallel text makes it difficult to reliably learn syntax-based translation rules. Focusing on syntax-based approaches within the language model might be a more direct way to address the poor grammatically of most machine translation output.

In previous work on syntax-based language modeling for machine translation, Wu and Wong (1998) included the score of an unlexicalized probabilistic context-free grammar (PCFG) in an ITG framework. Charniak et al. (2003) rescored a tree-to-string translation forest with a lexicalized parser and found more grammatical output despite a lower BLEU score (compared to an ngram version). More recently, Shen et al. (2008) used a trigram dependency model, with quite good results. Common to all of these approaches, though, is that target-side syntax is coupled with the translation channel.

This paper is distinct from previous work in that we decouple the structure of the target language tree from that of the synchronous grammar tree. Instead of extracting fragments of parse tree along with the phrase table and learning syntax-motivated reorderings, we start with simple phrase pairs and build the target-language structure at decoding time. We compare two syntax-based language models in the context of a fairly simple syntactic translation model, as a means of isolating the contributions and potential of these language models. The first is the statistical parser of Collins (1997) (Model 1). The second is based on the dependency parser of Klein and Manning (2004). Our translation model is a generic binary bracketing transduction grammar (Wu, 1997), whose main purpose is to restrict the alignment space to something that can be explored in polynomial time.

2 Motivation

From a grammatical standpoint, the output of most machine translation systems is very poor. Part of the reason may be that most systems do not incorporate syntax-based language models. The nominal task of the language model is to guide the search (decoding) procedure towards grammatical output, yet most systems use ngrams, which do not model sentence structure and cannot handle long-distance dependencies.

There are a number of reasons that researchers have stuck with ngrams. They work well, are easy to train, require no manual annotation, and are well-understood, with a long history in the speech recog-

nition community. Furthermore, the distinction between the duties of the language and translation models is somewhat dubious; the noisy-channel approach to MT permits a decomposition into these two models, but modern log-linear systems use many different component models with overlapping responsibilities. Most recent syntax-based systems put much of what might be considered the language model's responsibility into the translation model, by learning translation rules that produce structured target-language output from (flat) input phrases. An example is Galley et al. (2006), in which target language structures are linguistically motivated treebank parse fragments with an extended domain of locality, allowing, for example, a phrasal translation pair to specify for (optionally lexicalized) argument structure. In a different approach, Chiang (2007) used a generic (linguistically uninformed) grammar whose translation rules permitted lexicalized reorderings between the source and target languages. More recently, Shen et al. (2008) extended Chiang's system to learn pieces of dependency structure along with the rules.

These approaches (summarized in Figure 1) have shown significant progress, but there are reasons why we might want to construct target-side syntax independent of the translation channel. One problem faced by tree-to-string systems is syntactic divergence. Translation and its evaluation are not well-defined tasks; at a high level, we can make grammatical statements about languages, such as the observation that English is SVO and Japanese is SOV[1]. However, these general statements are not necessarily true of a particular sentence pair. Two sentences expressing the same semantics need not have corresponding structure, so it may be problematic to extract translation rules from them. Syntactic divergence is an even greater problem for language pairs involving a language with freer word order, which is one explanation for why syntax-based methods have done best when translating into English from fixed word-order languages like Chinese and French.

A second, related problem with syntax-based translation rules is reordering. While in theory using context (words in the Chiang model, and category combinations in Galley et al.'s) to license reordering can result in more informed reordering, it might also result in worse reordering or less reordering if the extracted rules are noisy or not general enough for the test data. A careful study of the various reordering models would be useful here, but empirical evaluation is another tool available to us.

[1] S = subject, O = object, V = verb.

TM	LM: syntax	LM: ngram
xRS	Marcu et al. (2006)	Galley et al. (2006) (*)
Hiero	Shen et al. (2008)	Chiang (2005)
BTG	This paper	Wu (1996)

Figure 1: Use of syntax in various systems. Galley et al. (2006) used no language model.

Finally, the problem is exacerbated by the dependence of structured translation rules on automatically parsed and automatically aligned data, which are noisy and ambiguous. Parsing accuracy, for example, is between ninety to ninenty-five per cent (for English), and parsing models do not generalize well across domains.

To that end, in this paper we examine two syntax-based language models in a weak translation model. Our approach decouples target-language trees from the translation rules, providing the language model with more freedom to express itself, and permitting us to isolate the strengths and weaknesses of the parser in selecting MT output.

3 Description

In this section we describe the parser models and the search algorithm. Our translation model is Dekai Wu's binary bracketing transduction grammar (BTG) with a single nonterminal X and rule weights learned automatically from Chinese-English data using EM. We extended our BTG implementation to use phrase pairs at the leaves instead of just word pairs. Our translation table is a joint distribution of Chinese / English phrase pairs of no more than seven words on either side.

Motivating our choice of BTG is a desire to assume as little about the space of reorderings as possible, and yet to have a principled alignment space that can be explored in polynomial time. Wellington et al. (2006) showed that word-to-word BTG can handle 95% of the alignments in their test data. Discontiguous alignments over small distances can be learned by memorizing the whole set

of phrasal pairs. One example is the well-known French/English translation pair {*ne X pas* / *do not X*}. BTG cannot handle this alignment, but it can compensate by learning instantiations of the phrase pair for different values of X.

It is worth pointing out explicitly that BTG in this form provides very little information to the translation process. In contrast to channel-rich models discussed earlier, BTG has only one parameter governing reordering, which expresses a preference for the straight binary rule over the inverted binary rule. This distortion model is even less informative than that of phrase-based systems such as Koehn et al. (2003)[2]. An uninformative translation model allows us to isolate the influence of the language model in assembling the translation hypotheses.

3.1 Parsing models

The task of a statistical parser is to assign probabilities to parse trees, that is, to provide a model of $\Pr(\pi|\mathbf{e})$, where π is a parse tree and $\mathbf{e}$ is an English sentence. Typically parsers are used to find the best parse, π^*, for a fixed input sentence $\mathbf{e}$, where $\pi^* = \operatorname{argmax}_\pi \Pr(\pi|\mathbf{e})$. Many parsers define generative models which assign joint probabilities $\Pr(\pi, \mathbf{e})$ to a (parse, sentence) pair, making use of the observation that

$$\operatorname*{argmax}_\pi \Pr(\pi|\mathbf{e}) = \operatorname*{argmax}_\pi \Pr(\pi, \mathbf{e})$$

A language model assigns probabilities $\Pr(\mathbf{e})$ to a sentence $\mathbf{e}$ in some language (in this paper, English). We can thus use a generative parser as a language model by marginalizing over all parse trees whose yield is our English sentence, computing $\Pr_{\mathtt{LM}}(\mathbf{e}) = \sum_\pi \Pr(\pi, \mathbf{e})$. In practice, we approximate the sum over all parses with the most probable parse, so that $\Pr_{\mathtt{LM}}(\mathbf{e}) \propto \max_\pi \Pr(\pi, \mathbf{e})$.

3.1.1 Collins Model 1

We chose Collins Model 1 as a natural starting point for experimenting with language models. It is not too difficult to implement and train, and its basic features – lexicalization and rule markovization – are at the core of the best generative parsers.

The exact model we used is the three-level backoff model given in Table 7.1 of Collins (1999), together with special handling of punctuation and conjuctions. This model comprises three distributions, one each for assigning probabilities to (a) head labels, (b) sibling labels and head tags, and (c) sibling head words.

3.1.2 Dependency parser

One potential problem with the use of the Collins parser is that it posits a great deal of hidden structure. The Treebank grammar, or the model parameterization used in the Collins parser, is not necessarily the most useful for machine translation; in fact, many researchers have shown that the treebank grammar isn't even the best choice for building parsers that are evaluated on their ability to recover treebank structure (Klein and Manning, 2003; Matsuzaki et al., 2005; Petrov and Klein, 2007). It stands to reason that a different grammar might do better as a language model for MT.

To explore this, we evaluated the dependency model of Klein and Manning (2004). This model imagines the generative process as adding left and right arguments to a lexical head word until a decision to stop is made, and then recursing on each of those arguments. We changed the model to use word-based (instead of tag-based) parameters, and backed off to unigram probabilities for the sibling distributions (we also ignored the constituent context model described in that paper).

3.2 Decoding with parsers

Our algorithm for decoding with parsers as language models is based on the algorithm for decoding with ngram language models, described in Wu (1997). That algorithm builds a dynamic programming chart in a bottom-up manner. Cells in the chart take the form of $[X, s, t, \mathbf{e}_l, \mathbf{e}_r]$, where X is the BTG nonterminal over span (s, t) of the input, and $\mathbf{e}_l, \mathbf{e}_r \in \mathcal{E}^*$ (where $\mathcal{E}$ is the target language vocabulary) are target-language boundary words which permit the ngram language model probability to be multiplied in when cells are combined.

In our algorithm, the dynamic programming state

[2]While in theory phrase-based systems should permit any reordering of the phrases between the source and target language sentence, many phrase-based systems turn off reordering entirely, or permit only position swaps between adjacent phrases. With these heuristics enabled, the set of alignments permitted is a strict subset of those attainable with BTG.

```
function DECODE(input sentence: c = c_{0..T})
    for span (s, t) of c in bottom-up order do
        for phrasal translation ē of c_{s..t} do
            F ← parse forest over ē
            add [s, t, F] to the chart
    for span (s, t) of c in bottom-up order do
        for split point S, s < S < t do
            for cell [s, S, F_1] in (s, S) do
                for cell [S, t, F_2] in (S, t) do
                    for BTG rule r do
                        if r.inverted then
                            F ← PARSE(F_2, F_1)
                        else
                            F ← PARSE(F_1, F_2)
                        add [s, t, F] to the chart
    return best cell in [0, T]
function PARSE(forest F_1, forest F_2)
    F.width ← F_1.width + F_2.width
    F.lchild = F_1
    F.rchild = F_2
    for span (u, v), 0 ≤ u < v ≤ F.width do
        if INTERSECTS(u, v, F_1.width) then
            for split point U, u < U < vt do
                (F_L, u_L, v_L) = CHART(F, u, U)
                (F_R, u_R, v_R) = CHART(F, U, v)
                for edge e_1 in (u_L, v_L) of F_L do
                    for edge e_2 in (u_R, v_R) of F_R do
                        parse with target grammar
                        put new edges in F
    return F
function CHART(forest F, u, v)
    if F.lchild ≠ NULL then
        U ← F.width
        if U ≥ v then
            return CHART(F.lchild, u, v)
        else if U ≤ u then
            return CHART(F.rchild, u − U, v − U)
    return (F, u, v)
function INTERSECTS(u, v, U)
    if U ≤ u or U ≥ v then return false
    return true
```

Figure 2: Algorithm.

takes the form $[s, t, \mathcal{F}]$[3], where $\mathcal{F}$ is the parse forest over the target language side. Using the actual forest as part of the dynamic programming state allows us to break the isomorphism between the synchronous and target-side grammars that is present in earlier syntax-based language modeling attempts (e.g., Charniak et al. (2003)). The algorithm, listed in Figure 2, works in the following manner. The input to the algorithm is a source-language sentence $\mathbf{c}$ of length T. We write $c_{s..t}$ to denote words $s+1..t$ in this sentence[4]. For each phrase $c_{s..t}$ on the input side, we consider some fixed number μ of phrasal translations $\mathbf{e}$. Each of these phrases is parsed with the target-language grammar, and the parse forest over the phrase becomes part of a new cell, which we place in the chart. The chart is thus seeded with no more than μT^2 cells.

Decoding proceeds by combining smaller cells to form larger ones. Our grammar has only two rules: the straight rule, and the inverted rule. To combine two cells $[s, S, \mathcal{F}_1]$ and $[S, t, \mathcal{F}_2]$, we place the parse forests $\mathcal{F}_1$ and $\mathcal{F}_2$ side by side in straight or inverted order, according to the rule under consideration, creating a new (incomplete) forest $\mathcal{F}$. We then complete this parse forest with the target language grammar using the CYK algorithm. One important consideration is that we only consider spans of that forest that cross the boundary position between $\mathcal{F}_1$ and $\mathcal{F}_2$, so as to avoid redundant parsing. All new edges formed are added to $\mathcal{F}$.

Note that merging forests in this manner results in an exponential waste of space, because lower forests are repeatedly copied and recopied higher and higher in the synchronous grammar tree. To avoid this, instead of actually copying $\mathcal{F}_1$ and $\mathcal{F}_2$ to $\mathcal{F}$, we maintain backpointers to the forests, and introduce a level of indirection. The CHART() method takes a starting forest and a target-language span (u, v), traversing its backpointers until it finds the chart from which edges over that span originated.

As an example of how the grammar structures are decoupled, consider decoding the Chinese sentence using the Collins parser language model:

0 预计 **1** 听 **2** 证 **3** 会 **4** 将 **5** 进行 **6** 两天 **7 · 8**

[3]Our synchronous grammar has only one nonterminal, so we will omit the X for notational convenience.

[4]Indices denote positions between words.

meaning *It is expected that the hearing will last for two days*. Figure 3 represents one path through the search algorithm. When synchronous parsing is complete, each cell over $(0, T)$ contains a complete parse forest over the target language side, like the one shown. From this forest, any structure over the target language can be extracted, regardless of the form of the synchronous grammar tree that produced it (which is monotonic in this example).

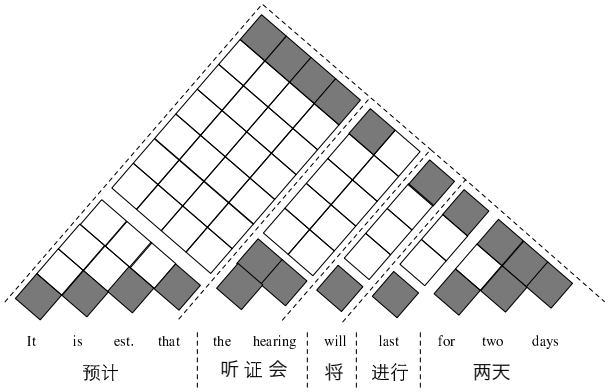


Figure 3: One path through the search algorithm. At the bottom of the tree are phrasal translations together with their parse forests. Darkened nodes represent constituents over those spans (the forest has been simplified for presentation purposes, as the actual forest would have many target-grammar nonterminals over each span). The dashed lines denote the order in which the cells are combined; each application of a binary synchronous grammar rule concatenates two forests and produces a new set of edges that complete the forest.

3.3 Computational complexity

Searching for the best sentence under a CFG-based language model and a CFG-based translation model is an instance of the CFG intersection problem, which is undecidable in general (Hopcroft and Ullman, 1979). However, the nature of the intersection problem handled by our algorithm is somewhat easier, because the CFG derived from the synchronous grammar in our situation is non-recursive. This grammar is formed in the following manner on a per-sentence basis from the dynamic programming chart built from the synchronous grammar G and the source language sentence $c_{1..T}$. Each cell $[X, s, t]$ in this chart was built from smaller cells $[Y, s, S]$ and $[Z, S, t]$ with ITG rule $X \rightarrow [Y\ Z] \in G$; from each of these, we will have a rule of the form ${}_sX_t \rightarrow {}_sY_S\ \ {}_SZ_t$. Although an ITG grammar can be recursive, rules formed from the chart in this manner are nonrecursive because we disallow insertions, meaning that each rule must consume part of the input. Unfortunately, even the problem of finding a string in the intersection of two non-recursive context-free grammars (or one non-recursive and one recursive grammar, as in our case) is PSPACE-complete (Nederhof and Satta, 2004), making it unlikely that an efficient algorithm exists for finding the optimal translation under our model. Taking μ again to be the number of phrasal translation candidates considered for each span of the input, the actual complexity of the algorithm in Figure 2 is $\mathcal{O}(T^3\mu^{2T}P(T))$. The T^3 term is the time required to enumerate the spans and split points of the input, the second term is the number of target-language forest merges ($\mu \cdot \mu$ squared repeatedly to the $\mathcal{O}(T)$ height of a binary tree), and $P(T)$ is the complexity of parsing in the target language grammar.

3.4 Scoring and pruning

It is clear from the complexity results that the search space for our decoder is prohibitively large. In light of this, we experimented extensively with pruning. To begin with, we use a two-pass coarse-to-fine approach. For the coarse pass, we built a chart with a bigram language model and computed the inside and outside probabilities for each cell in that chart. In the fine pass, we ignore any cell whose coarse inside-outside score is below some constant times the maximum inside-outside score in the coarse chart.

Another difficulty is in computing the score of a parse forest. Under our model, the score of a cell in the chart is a log-linear combination of features, one of which is the target-language parse forest associated with the cell. There are good ways to score edges in this forest, but it is not easy to rate the forest itself. For this paper, we took a parse forest's score to be the sum of the best edge over each span of the forest, divided by the number of spans. An obvious step for future work would be to apply a coarse-to-fine procedure to the target-language forests.

4 Experiments

We used in-house implementations of the parsers and the BTG decoder, all written in C++.

Our phrase translation table came from two sources. The first is 44 million word pairs trained on 160 million words of newswire text. We supplemented this source with a phrase-to-phrase table learned from about 833K parallel sentences of newswire text (the majority of it from Munteanu and Marcu (2005)). We trained the phrases by running GIZA++ in both directions, taking the union of the resulting alignments, and then extracting all phrase pairs consistent with the unioned alignment and fewer than eight words on either side, along with the counts. Probabilities were set to relative frequency. We then combined the two tables by linearly interpolating them, experimenting with different weights until we found one that produced the best results on the development set.

The ngram language model was trained on the English side of this parallel corpus using SRILM (Stolcke, 2002). The parsers were trained the 49,208 trees from the Penn Treebank plus parses of most of the English side of our parallel data, which was automatically parsed with the parser of Charniak (2000). With both parsers, we treated commas, colons, and periods as if they were just regular words in the vocabulary. Quotation marks were treated as an unknown word, and we used the same set of fifty bins (computed based on word surface features) for unknown words used in Petrov et al. (2006).

Our development data consisted of all Chinese sentences with twenty or fewer words from the NIST 2002 evaluation (371 sentences). The parameters of our log-linear model (which includes weights for the ngram and parser models, along with a length bonus) were set with a hill-climbing procedure on the development data. These parameters were then used to produce results from our test data, which was the portion of the NIST 2003 evaluation dataset with no more than twenty words (347 sentences). Our evaluation metric was case-insensitive BLEU-4 (Papineni et al., 2002). Table 1 contains the results of runs on the development and test sets.

RUN	DEV/10	DEV/4	TEST/4
No LM	17.35	13.12	15.41
Bigram	25.09	19.18	18.62
Trigram	26.18	20.08	21.55
Collins + bigram	25.00	18.92	18.13
Dep + bigram	24.49	18.73	18.52

Table 1: Results (BLEU-4 scores). The number following the slash indicates the number of human references used in computing the BLEU score. No post-processing was applied to the MT output. The weight for the parser score was only allowed to go as low at 0.1, which is why the parser + bigram models are able to score slightly below the bigram model alone.

5 Discussion

All of our syntax-based language models underperformed our ngram baselines, suggesting that unadapted parsers with a weak translation model under our search approach are not well-suited to the machine translation task. However, there is much to be learned from analysis of the strengths and weaknesses of the parsers that will be useful for developing language models for machine translation.

5.1 Search error vs. model error

A question that immediately comes to mind is whether the low BLEU scores are due to parsers being poor language models, or to search errors resulting from the extensive pruning. This question is difficult to answer, because without pruning the decoding cannot be completed in a feasible amount of time and space. We can cut back on pruning for a (time-consuming) single run across the development corpus, but the hill-climbing procedure to find the best set of model parameters requires many runs.

We explore this issue with two smaller experiments designed to test the parsers in more limited settings.

(1) *Shuffling*. The first experiment was designed to test how well each language model performs on the reordering task alone, without having to worry about word and phrase selection. We took the 1,464 sentences with no more than twenty-five words from section twenty-three of the WSJ portion of the Penn Treebank and produced three random ITG permutations of each[5]. We then parsed these sentences with our BTG decoder using an identity word translation table that translated each word as itself, scoring cells in the chart with the language model only.

[5]By which we mean a permutation that is attainable under the ITG constraints. To select the permutation, we recursively choose a split point and decide whether to invert the two halves.

Note that for this task, we removed the WSJ portion of the Treebank from the parser training data, so that parsers were retrained on only our automatically parsed data. We also applied the parser language models directly, skipping the coarse step used in the decoding results above.

We scored each sentence by summing the distances from each word in the LM-restored sentence to its position in the original unpermuted sentence. The results can be found in Table 2. They show that the Collins parser does the best job of restoring the original sentence order.

distance	baseline	trigram	collins	depend
0	67	249	**325**	142
≤ 25	909	1038	**1223**	962
≤ 50	1690	1735	**1945**	1706
≤ 75	2282	2282	**2507**	2353
≤ 100	2800	2807	**3037**	2890
≤ 125	3138	3231	**3424**	3296
≤ 150	3460	3600	**3747**	3666
≤ 175	3699	3857	**3977**	3902
≤ 200	3910	4017	**4158**	4099
≤ 225	4085	4215	**4264**	4222
≤ 250	4207	4326	**4330**	4311
≤ 275	4290	**4376**	4362	4359
≤ 300	4364	**4392**	4375	4385
≤ 325	**4392**	**4392**	4378	4389
≤ 350	**4392**	**4392**	**4392**	**4392**

Table 2: Shuffling experiment: number of sentences (from 4,392 total) having various edit distances, for each language model.

(2) *Selection.* In a second experiment, we test how well each language model does at selecting among the translation candidates on a monotonic translation task. Limiting the number of phrasal translation candidates for each input phrase to one, our system produces a BLEU score of 18.78 with the bigram decoder and 18.30 for the Collins decoder (once again, no coarse pass is used).

In summary, the Collins parser did better than the trigram model on the reordering task, but worse on the selection task. Note that even in these more limited experiments, extensive pruning is still required, so the better performance of the Collins parser on this first task is significant. One possible explanation that distinguishes these cases is that in the first case there is some grammatical permutation of the input, which is not necessarily the case in the second experiment. The Collins parser was designed to distinguish among structures over input whose grammaticality is assumed, whereas in MT most (if not all) of the possible translations will not be grammatical.

5.2 Sentence-level performance

Although the overall BLEU scores of the parser-based systems were lower, we wondered what was happening at the sentence level. Table 3 shows a breakdown in per-sentence performance of each of the models on the development set.

LM	best	xbest
baseline	70	53
bigram	127	39
trigram	137	82
Collins + bigram	122	45
depend + bigram	117	45

Table 3: Number of times each system shared the best sentence-level BLEU score (best) and had the exclusively best score (xbest).

To compute this number, we used a smoothed BLEU score similar to that of Liang et al. (2006), which will only be zero if no ngrams (for all of $1 \leq n \leq 4$) were matched.

In light of the other results, it is unsurprising that the trigram model outperforms all of the other models. It is interesting, however, to compute the best BLEU score attainable by using an oracle to select the best sentence (see Table 4). The parser-based

systems	BLEU
All five	30.29
Trigram + bigram	28.42
Trigram + Collins + depend	29.23

Table 4: Oracle selection of output across different sets of system output.

systems are capable of improving the BLEU score over the ngram language models, but the model is unable to make use of it. One possible explanation for this that the decoding model is very coarse, incorporating the parser scores with a single weight. Adding finer-level weights in a generative framework presents difficulties for the training procedure.

Discriminative approaches may be a good way to address this problem.

5.3 Fluency

One observation from a manual inspection of the results is that the fluency of the output from the parser-enabled decoders is poor. This, despite the fact that a good high-level structure is often present. Consider a Chinese sentence, one reference for which is *I think it's just an internal affair of the US*. The trigram decoder produces the passable *I think this issue is totally the United States*. The Collins and dependency outputs can be found in Figure 4. In both cases, part of the problem is the phrase *the this* (highlighted above). Both parsers are led astray because of noise in the translation table. The Chinese character translated as *the* in both cases has the following top five translation candidates:

-7.55655 influence in the
-9.46839 influence in
-9.77855 impact on
-10.6608 influence
-11.1026 the

This high-probability noise in the translation table overwhelms each of the parsers, but for different reasons. Although neither parsing model saw *the* in the training data as a direct argument of the head that selects them in the above parses, the backoff model of each allows them to choose it in this situation with a reasonable probability. For the Collins parser, the overall structure is convincing enough that it can overlook this low-probability NP; for the dependency parser, the combination is an accident of the independent manner of argument generation. In contrast, the trigram model rules out this sequence quite easily.

This situation highlights the fact that ngrams are important for ensuring fluency at a local level, and is likely that they will be useful as a complement of grammatical language models accounting for long-distance dependencies.

5.4 Consistency of grammatical role

Homographs present another problem to the parsers. Distributions over argument structure vary widely depending on the grammatical function of a word, but many words share a form that takes on multiple grammatical roles. One example is the

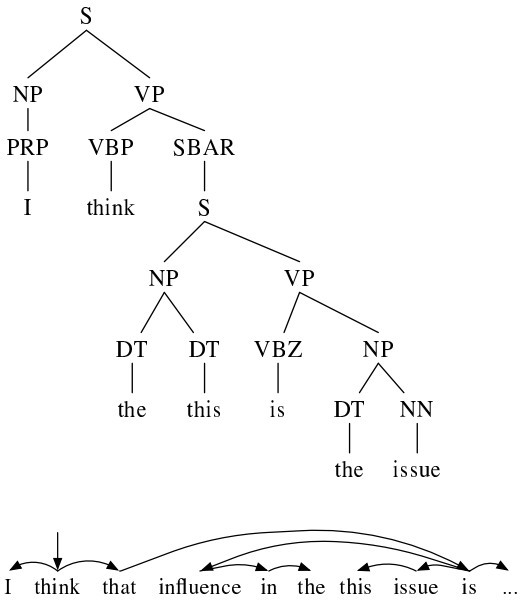
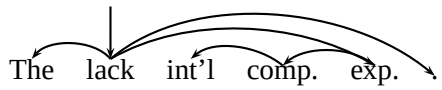
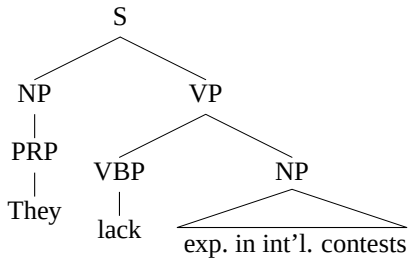


Figure 4: Collins and dependency decoder outputs from a development set sentence.

word *lack*, which can be either a noun or a verb. Consider another Chinese sentence that is translated as *They lack experience in international competitions*. Our dependency decoder produces the following translation for this sentence:



and the Collins decoder produces this one:



The raw counts for *lack* as a noun are much higher in our training data than for *lack* as a verb. Thus the dependency parser, which does not account for the verb/noun distinction, prefers *the* to *they* as a left argument by nearly 20 to 1. In the Collins model, where the distinction is primary, *they* is the most probable left argument.

The decoder's mistaken treatment of *lack* as a noun is an unavoidable problem; even with the

Collins parser, there will likely be a hypothesis in the chart positing the noun form of the word. The more serious problem here is the dependency decoder's inconsistent treatment of the word: its left argument casts *lack* as a noun, while its right argument treats it as a verb. Moreover, the word's choice as the dependency model's root is licensed mostly by the verb forms of *lack* found in the Treebank.

This isn't necessarily an argument for an explicitly defined notion of category. It may be that word category and consistency can be enforced by doing away with the modeling assumption that sibling arguments are generated independently of one another, or some other such method.

5.5 Parameterization

As we have mentioned earlier, the parameterizations of the parsing models (in terms of independence assumptions and backoff models) discussed in this paper were chose to maximize performance against the labeled recall and precision tasks of parsing. What's more, parsers assume their input to be grammatical. In contrast, the task for SBLMs for MT is to impose grammaticality on what can be roughly viewed as a bag of words. Whereas a parser may rule out a lexicalized constituent, say because it cannot produce one of its arguments with high probability, in translation there is much more freedom to find what the language model needs, especially if our reordering model does not impose many restrictions.

The examples above suggest what may be some of the problems – in particular, the fact that parsers generate arguments independently of each other and with very limited notions of valence.

6 Summary & Conclusion

In this paper we have presented an algorithm that decouples the syntax of target language structure from the reordering model, and thus can be used with any phrase-based translation model. Furthermore, we have investigated the use of existing parsing technology as syntax-based language models for machine translation. In so doing, we have discovered a number of situations where parsers fail, which we hope will inform grammatical language models more suited to the needs of language modeling for MT. Our analysis suggests that key aspects of syntax-based language models for MT will include tightening or removing the independence assumptions of parsing models, maintaining some notion of category, and employing a finer-grained set of weights over rules and categories.

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

References

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for machine translation. In *Proc. MT Summit IX*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the ACL (NAACL)*, pages 132–139, Seattle, Washington.

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

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL/EACL-97*, pages 16–23, Madrid, Spain.

Michael John Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

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 COLING/ACL-06*, pages 961–968, July.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL-03*, pages 423–430.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proceedings of ACL-04*, page 478, Morristown, NJ, USA. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL-03*, Edmonton, Alberta.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia, July.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of ACL-05*, pages 75–82.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

Mark-Jan Nederhof and Giogio Satta. 2004. The language intersection problem for non-recursive context-free grammars. *Information and Computation*, 192:172–184.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL-02*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, April.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, Columbus, OH. ACL.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of COLING/ACL-06*.

Dekai Wu and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *COLING/ACL-98*.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *34th Annual Meeting of the Association for Computational Linguistics*.

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