Large-Scale Syntactic Language Modeling with Treelets

Adam Pauls **Dan Klein**
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720, USA
{adpauls,klein}@cs.berkeley.edu

Abstract

We propose a simple generative, syntactic language model that conditions on overlapping windows of tree context (or *treelets*) in the same way that n-gram language models condition on overlapping windows of linear context. We estimate the parameters of our model by collecting counts from automatically parsed text using standard n-gram language model estimation techniques, allowing us to train a model on over one billion tokens of data using a single machine in a matter of hours. We evaluate on perplexity and a range of grammaticality tasks, and find that we perform as well or better than n-gram models and other generative baselines. Our model even competes with state-of-the-art discriminative models hand-designed for the grammaticality tasks, despite training on positive data alone. We also show fluency improvements in a preliminary machine translation experiment.

1 Introduction

N-gram language models are a central component of all speech recognition and machine translation systems, and a great deal of research centers around refining models (Chen and Goodman, 1998), efficient storage (Pauls and Klein, 2011; Heafield, 2011), and integration into decoders (Koehn, 2004; Chiang, 2005). At the same time, because n-gram language models only condition on a local window of linear word-level context, they are poor models of long-range syntactic dependencies. Although several lines of work have proposed generative syntactic language models that improve on n-gram models for moderate amounts of data (Chelba, 1997; Xu et al., 2002; Charniak, 2001; Hall, 2004; Roark, 2004), these models have only recently been scaled to the impressive amounts of data routinely used by n-gram language models (Tan et al., 2011).

In this paper, we describe a generative, syntactic language model that conditions on local context treelets[1] in a parse tree, backing off to smaller treelets as necessary. Our model can be trained simply by collecting counts and using the same smoothing techniques normally applied to n-gram models (Kneser and Ney, 1995), enabling us to apply techniques developed for scaling n-gram models out of the box (Brants et al., 2007; Pauls and Klein, 2011). The simplicity of our training procedure allows us to train a model on a billion tokens of data in a matter of hours on a single machine, which compares favorably to the more involved training algorithm of Tan et al. (2011), who use a two-pass EM training algorithm that takes several days on several hundred CPUs using similar amounts of data.

The simplicity of our approach also contrasts with recent work on language modeling with tree substitution grammars (Post and Gildea, 2009), where larger treelet contexts are incorporated by using sophisticated priors to learn a segmentation of parse trees. Such an approach implicitly assumes that a "correct" segmentation exists, but it is not clear that this is true in practice. Instead, we build upon the success of n-gram language models, which do not assume a segmentation and instead score all overlapping contexts.

We evaluate our model in terms of perplexity, and show that we achieve the same performance as a state-of-the-art n-gram model. We also evaluate our model on several grammaticality tasks proposed in

[1]We borrow the term *treelet* from Quirk et al. (2005), who use it to refer to an arbitrary connected subgraph of a tree.

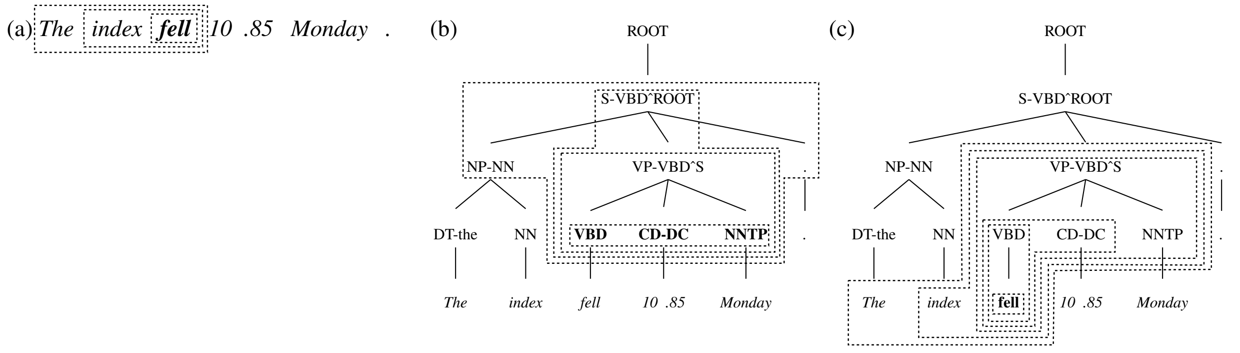

Figure 1: Conditioning contexts and back-off strategies for Markov models. The bolded symbol indicates the part of the tree/sentence being generated, and the dotted lines represent the conditioning contexts; back-off proceeds from the largest to the smallest context. (a) A trigram model. (b) The context used for non-terminal productions in our treelet model. For this context, P=VP-VBD^S, P'=S-VBD^ROOT, and r'=S-VBD^ROOT→NP-NN VP-VBD^S . (c) The context used for terminal productions in our treelet model. Here, P=VBD, R=CD-DC, r'=VP-VBD^S→VBD CD-DC NNTP, w_{-1}=*index*, and w_{-2}=*The*. Note that the tree is a modified version of a standard Penn Treebank parse – see Section 3 for details.

the literature (Okanohara and Tsujii, 2007; Foster et al., 2008; Cherry and Quirk, 2008) and show that it consistently outperforms an n-gram model as well as other head-driven and tree-driven generative baselines. Our model even competes with state-of-the-art discriminative classifiers specifically designed for each task, despite being estimated on positive data alone. We also show fluency improvements in a preliminary machine translation reranking experiment.

2 Treelet Language Modeling

The common denominator of most n-gram language models is that they assign probabilities roughly according to empirical frequencies for observed n-grams, but fall back to distributions conditioned on smaller contexts for unobserved n-grams, as shown in Figure 1(a). This type of smoothing is both highly robust and easy to implement, requiring only the collection of counts from data.

We would like to apply the same smoothing techniques to distributions over rule yields in a constituency tree, conditioned on contexts consisting of previously generated treelets (rules, nodes, etc.). Formally, let T be a constituency tree consisting of context-free rules of the form $r = P \rightarrow C_1 \cdots C_d$, where P is the parent symbol of rule r and $C_1^d = C_1 \ldots C_d$ are its children. We wish to assign probabilities to trees[2]

$$p(T) = \prod_{r \in T} p(C_1^d | h)$$

where the conditioning context h is some portion of the already-generated parts of the tree. In this paper, we assume that the children of a rule are expanded from left to right, so that when generating the yield C_1^d, all treelets above and left of the parent P are available. Note that a raw PCFG would condition only on P, i.e. $h = P$.

As in the n-gram case, we would like to pick h to be large enough to capture relevant dependencies, but small enough that we can obtain meaningful estimates from data. We start with a straightforward choice of context: we condition on P, as well as the rule r' that generated P, as shown in in Figure 1(b).

Conditioning on the parent rule r' allows us to capture several important dependencies. First, it captures both P and its parent P', which predicts the distribution over child symbols far better than just P (Johnson, 1998). Second, it captures positional effects. For example, subject and object noun phrases (NPs) have different distributions (Klein and Manning, 2003), and the position of an NP relative to a verb is a good indicator of this distinction. Finally, the generation of words at preterminals can condition on siblings, allowing the model to capture, for example, verb subcategorization frames.

We should be clear that we are not the first

[2]A distribution over trees also induces a distribution over sentences w_1^ℓ given by $p(w_1^\ell) = \sum_{T:s(T)=w_1^\ell} p(T)$, where $s(T)$ is the terminal yield of T.

to use back-off-based smoothing for syntactic language modeling – such techniques have been applied to models that condition on head-word contexts (Charniak, 2001; Roark, 2004; Zhang, 2009). Parent rule context has also been employed in translation (Vaswani et al., 2011). However, to our knowledge, we are the first to apply these techniques for language modeling on large amounts of data.

2.1 Lexical context

Although it is tempting to think that we can replace the left-to-right generation of n-gram models with the purely top-down generation of typical PCFGs, in practice, words are often highly predictive of the words that follow them – indeed, n-gram models would be terrible language models if this were not the case. To capture linear effects, we extend the context for terminal (lexical) productions to include the previous two words w_{-2} and w_{-1} in the sentence in addition to r'; see Figure 1(c) for a depiction. This allows us to capture collocations and other lexical correlations.

2.2 Backing off

As with n-gram models, counts for rule yields conditioned on r' are sparse, and we must choose an appropriate back-off strategy. We handle terminal and non-terminal productions slightly differently.

For non-terminal productions, we back off from r' to P and its parent P', and then to just P. That is, we back off from a rule-annotated grammar $p(C_1^d|P, P', r')$ to a parent-annotated grammar (Johnson, 1998) $p(C_1^d|P, P')$, then to a raw PCFG $p(C_1^d|P)$. In order to generalize to unseen rule yields C_1^d, we further back off from the basic PCFG probability $p(C_1^d|P)$ to $p(C_i|C_{i-3}^{i-1}, P)$, a 4-gram model over symbols C conditioned on P, interpolated with an unconditional 4-gram model $p(C_i|C_{i-3}^{i-1})$. In other words, we back off from a raw PCFG to

$$\lambda \prod_{i=1}^{d} p(C_i|C_{i-3}^{i-1}, P) + (1-\lambda) \prod_{i=1}^{d} p(C_i|C_{i-3}^{i-1})$$

where $\lambda = 0.9$ is an interpolation constant.

For terminal (i.e lexical) productions, we first remove lexical context, backing off from $p(w|P, R, r', w_{-1}, w_{-2})$ to $p(w|P, R, r', w_{-1})$ and then $p(w|P, R, r')$. From there, we back off to $p(w|P, R)$ where R is the sibling immediately to the right of P, then to a raw PCFG $p(w|P)$, and finally to a unigram distribution. We chose this scheme because $p(w|P, R)$ allows, for example, a verb to be generated conditioned on the non-terminal category of the argument it takes (since arguments usually immediately follow verbs). We depict these two back-off schemes pictorially in Figure 1(b) and (c).

2.3 Estimation

Estimating the probabilities in our model can be done very simply using the same techniques (in fact, the same code) used to estimate n-gram language models. Our model requires estimates of four distributions: $p(C_1^d|P, P', r')$, $p(w|P, R, r', w_{-1}, w_{-2})$, $p(C_i|C_{i-n+1}^{i-1}, P)$, and $p(C_i|C_{i-n+1}^{i-1})$. In each case, we require empirical counts of treelet tuples in the same way that we require counts of word tuples for estimating n-gram language models.

There is one additional hurdle in the estimation of our model: while there exist corpora with human-annotated constituency parses like the Penn Treebank (Marcus et al., 1993), these corpora are quite small – on the order of millions of tokens – and we cannot gather nearly as many counts as we can for n-grams, for which billions or even trillions (Brants et al., 2007) of tokens are available on the Web. However, we can use one of several high-quality constituency parsers (Collins, 1997; Charniak, 2000; Petrov et al., 2006) to automatically generate parses. These parses may contain errors, but not all parsing errors are problematic for our model, since we only care about the sentences generated by our model and not the parses themselves. We show in our experiments that the addition of data with automatic parses does improve the performance of our language models across a range of tasks.

3 Tree Transformations

In the previous section, we described how to condition on rich parse context to better capture the distribution of English trees. While such context allows our model to capture many interesting dependencies, several important dependencies require additional attention. In this section, we describe a

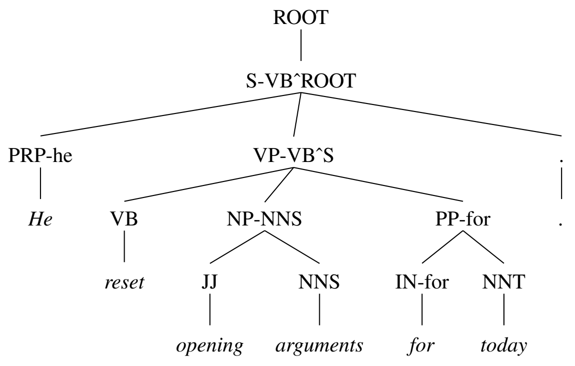

Figure 2: A sample parse from the Penn Treebank after the tree transformations described in Section 3. Note that we have not shown head tag annotations on preterminals because in that case, the head tag is the preterminal itself.

number of transformations of Treebank constituency parses that allow us to capture such dependencies. We list the annotations and deletions in the order in which they are performed. A sample transformed tree is shown in Figure 2.

Temporal NPs Following Klein and Manning (2003), we attempt to annotate temporal noun phrases. Although the Penn Treebank annotates temporal NPs, most off-the-shelf parsers do not retain these tags, and we do not assume their presence. Instead, we mark any noun that is the head of a NP-TMP constituent at least once in the Treebank as a temporal noun, so for example *today* would be tagged as NNT and *months* would be tagged as NNTS.

Head Annotations We annotate every non-terminal or preterminal with its head word if the head is a closed-class word[3] and with its head tag otherwise. Klein and Manning (2003) used head tag annotation extensively, though they applied their splits much more selectively.

NP Flattening We delete NPs dominated by other NPs, unless the child NPs are in coordination or apposition. These NPs typically occur when nouns are modified by PPs, as in (NP (NP (NN stock) (NNS sales)) (PP (IN by) (NNS traders))). By removing the dominated NP, we allow the production NNS→*sales* to condition on the presence of a modifying PP (here a PP head-annotated with *by*).

Number Annotations Numbers are divided into five classes: CD-YR for numbers that consist of four digits (which are usually years); CD-NM for entirely numeric numbers; CD-DC for numbers that have a decimal; CD-MX for numbers that mix letters and digits; and CD-AL for numbers that are entirely alphabetic.

[3]We define the following to be closed class words: any punctuation; all inflections of the verbs *do*, *be*, and *have*; and any word tagged with IN, WDT, PDT, WP, WP$, TO, WRB, RP, DT, SYM, EX, POS, PRP, AUX, or CC.

SBAR Flattening We remove any sentential (S) nodes immediately dominated by an SBAR. S nodes under SBAR have very distinct distributions from other sentential nodes, mostly due to empty subjects and/or objects.

VP Flattening We remove any VPs immediately dominating a VP, unless it is conjoined with another VP. In the Treebank, chains of verbs (e.g. *will be going*) have a separate VP for each verb. By flattening such structures, we allow the main verb and its arguments to condition on the whole chain of verbs. This effect is particularly important for passive constructions.

Gapped Sentence Annotation Collins (1999) and Klein and Manning (2003) annotate nodes which have empty subjects. Because we only assume the presence of automatically derived parses, which do not produce the empty elements in the original Treebank, we must identify such elements on our own. We use a very simple procedure: we annotate all S or SBAR nodes that have a VP before any NPs.

Parent Annotation We annotate all VPs with their parent symbol. Because our treelet model already conditions on the parent, this has the effect of allowing verbs to condition on their grandparents. This was important for VPs under SBAR nodes, which often have empty objects. We also parent-annotated any child of the ROOT.

Unary Deletion We remove all unary productions except the root and preterminal productions, keeping only the bottom-most symbol. Because we are not interested in the internal labels of the trees, unaries are largely a nuisance, and their removal brings many symbols into the context of others.

4 Scoring a Sentence

Computing the probability of a sentence w_1^ℓ under our model requires summing over all possible parses of w_1^ℓ. Although our model can be formulated as a straightforward PCFG, allowing $O(\ell^3)$ computation of this sum, the grammar constant for this PCFG would be unmanageably large (since every parent rule context would need its own state in the grammar), and extensive pruning would be in order.

In practice, however, language models are normally integrated into a decoder, a non-trivial task that is highly problem-dependent and beyond the scope of this paper. For machine translation, a model that builds target-side constituency parses, such as that of Galley et al. (2006), combined with an efficient pruning strategy like cube pruning (Chiang,

5-GRAM
The board 's will soon be feasible , from everyday which Coke 's cabinet hotels .
They are all priced became regulatory action by difficulty caused nor Aug. 31 of Helmsley-Spear :
Lakeland , it may take them if the 46-year-old said the loss of the Japanese executives at him :
But 8.32 % stake in and Rep. any money for you got from several months , " he says .
TREELET
Why a $ 1.2 million investment in various types of the bulk of TVS E. August ?
" One operating price position has a system that has Quartet for the first time , " he said .
He may enable drops to take , but will hardly revive the rush to develop two-stroke calculations . "
The centers are losses of meals , and the runs are willing to like them .

Table 1: The first four samples of length between 15 and 20 generated from the 5-GRAM and TREELET models.

2005), should be able to integrate our model without much difficulty.

That said, for evaluation purposes, whenever we need to query our model, we use the simple strategy of parsing a sentence using a black box parser, and summing over our model's probabilities of the 1000-best parses.[4] Note that the bottleneck in this case is the parser, so our model can essentially score a sentence at the speed of a parser.

5 Experiments

We evaluate our model along several dimensions. We first show some sample generated sentences in Section 5.1. We report perplexity results in Section 5.2. In Section 5.3, we measure its ability to distinguish between grammatical English and various types of automatically generated, or *pseudo-negative*,[5] English. We report machine translation reranking results in Section 5.4.

5.1 Generating Samples

Because our model is generative, we can qualitatively assess it by generating samples and verifying that they are more syntactically coherent than other approaches. In Table 1, we show the first four samples of length between 15 and 20 generated from our model and a 5-gram model trained on the Penn Treebank.

[4]We found that using the 1-best worked just as well as the 1000-best on our grammaticality tasks, but significantly overestimated our model's perplexities.

[5]We follow Okanohara and Tsujii (2007) in using the term pseudo-negative to highlight the fact that automatically generated negative examples might not actually be ungrammatical.

5.2 Perplexity

Perplexity is the standard intrinsic evaluation metric for language models. It measures the inverse of the per-word probability a model assigns to some held-out set of grammatical English (so lower is better). For training data, we constructed a large treebank by concatenating the WSJ and Brown portions of the Penn Treebank, the 50K BLLIP training sentences from Post (2011), and the AFP and APW portions of English Gigaword version 3 (Graff, 2003), totaling about 1.3 billion tokens. We used the human-annotated parses for the sentences in the Penn Treebank, but parsed the Gigaword and BLLIP sentences with the Berkeley Parser. Hereafter, we refer to this training data as our 1B corpus. We used Section 0 of the WSJ as our test corpus. Results are shown in Table 2. In addition to our TREELET model, we also show results for the following baselines:

5-GRAM A 5-gram interpolated Kneser-Ney model.

PCFG-LA The Berkeley Parser in language model mode.

HEADLEX A head-lexicalized model similar to, but more powerful[6] than, Collins Model 1 (Collins, 1999).

PCFG A raw PCFG.

TREELET-TRANS A PCFG estimated on the trees after the transformations of Section 3.

TREELET-RULE The TREELET-TRANS model with the parent rule context described in Section 2. This is equivalent to the full TREELET model without the lexical context described in Section 2.1.

[6]Specifically, like Collins Model 1, we generate a rule yield conditioned on parent symbol P and head word h by first generating its head symbol C_h, then generating the head words and symbols for left and right modifiers outwards from C_h. Unlike Model 1, which generates each modifier head and symbol conditioned only on C_h, h, and P, we additionally condition on the previously generated modifier's head and symbol and back off to Model 1.

Model	Perplexity
PCFG	1772
TREELET-TRANS	722
TREELET-RULE	329
TREELET	**198**†
PCFG-LA	330**
HEADLEX	299
5-GRAM	207†

Table 2: Perplexity of several generative models on Section 0 of the WSJ. The differences between scores marked with † are not statistically significant. PCFG-LA (marked with **) was only trained on the WSJ and Brown corpora because it does not scale to large amounts of data.

We used the Berkeley LM toolkit (Pauls and Klein, 2011), which implements Kneser-Ney smoothing, to estimate all back-off models for both n-gram and treelet models. To deal with unknown words, we use the following strategy: after the first 10000 sentences, whenever we see a new word in our training data, we replace it with a signature[7] 10% of the time.

Our model outperforms all other generative models, though the improvement over the n-gram model is not statistically significant. Note that because we use a k-best approximation for the sum over trees, all perplexities (except for PCFG-LA and 5-GRAM) are pessimistic bounds.

5.3 Classification of Pseudo-Negative Sentences

We make use of three kinds of automatically generated pseudo-negative sentences previously proposed in the literature: Okanohara and Tsujii (2007) proposed generating pseudo-negative examples from a trigram language model; Foster et al. (2008) create "noisy" sentences by automatically inserting a single error into grammatical sentences with a script that randomly deletes, inserts, or misspells a word; and Och et al. (2004) and Cherry and Quirk (2008) both use the 1-best output of a machine translation system. Examples of these three types of pseudo-negative data are shown in Table 3. We evaluate our model's ability to distinguish positive from pseudo-negative data, and compare against generative baselines and state-of-the-art discriminative methods.

[7]We use signatures generated by the Berkeley Parser. These signatures capture surface features such as capitalization, presents of digits, and common suffixes. For example, the word *vexing* would be replaced with the signature UNK-ing.

Noisy	*There was were many contributors.*
Trigram	*For years in dealer immediately .*
MT	*we must further steps .*

Table 3: Sample pseudo-negative sentences.

We would like to use our model to make grammaticality judgements, but as a generative model it can only provide us with probabilities. Simply thresholding generative probabilities, even with a separate threshold for each length, has been shown to be very ineffective for grammaticality judgements, both for n-gram and syntactic language models (Cherry and Quirk, 2008; Post, 2011). We used a simple measure for isolating the syntactic likelihood of a sentence: we take the log-probability under our model and subtract the log-probability under a unigram model, then normalize by the length of the sentence.[8] This measure, which we call the *syntactic log-odds ratio* (SLR), is a crude way of "subtracting out" the semantic component of the generative probability, so that sentences that use rare words are not penalized for doing so.

5.3.1 Trigram Classification

To facilitate comparison with previous work, we used the same negative corpora as Post (2011) for trigram classification. They randomly selected 50K train, 3K development, and 3K positive test sentences from the BLLIP corpus, then trained a trigram model on 450K BLLIP sentences and generated 50K train, 3K development, and 3K negative sentences. We parsed the 50K positive training examples of Post (2011) with the Berkeley Parser and used the resulting treebank to train a treelet language model. We set an SLR threshold for each model on the 6K positive and negative development sentences.

Results are shown in Table 4. In addition to our generative baselines, we show results for the discriminative models reported in Cherry and Quirk (2008) and Post (2011). The former train a latent PCFG support vector machine for binary classification (LSVM). The latter report results for two binary classifiers: RERANK uses the reranking features of Charniak and Johnson (2005), and TSG uses

[8]Och et al. (2004) also report using a parser probability normalized by the unigram probability (but not length), and did not find it effective. We assume this is either because the length-normalization is important, or because their choice of syntactic language model was poor.

Generative		
	BLLIP	**1B**
PCFG	81.5	81.8
TREELET-TRANS	87.7	90.1
TREELET-RULE	89.8	**94.1**
TREELET	88.9	93.3
PCFG-LA	87.1*	–
HEADLEX	87.6	92.0
5-GRAM	67.9	87.5
Discriminative		
	BLLIP	**1B**
LSVM	81.42**	–
TSG	89.9	–
RERANK	**93.0**	–

Table 4: Classification accuracy for trigram pseudo-negative sentences on the BLLIP corpus. The number reported for PCFG-LA is marked with a * to indicate that this model was trained on the training section of the WSJ, not the BLLIP corpus. The number reported for LSVM (marked with **) was evaluated on a different random split of the BLLIP corpus, and so is not directly comparable.

indicator features extracted from a tree substitution grammar derivation of each sentence.

Our TREELET model performs nearly as well as the TSG method, and substantially outperforms the LSVM method, though the latter was not tested on the same random split. Interestingly, the TREELET-RULE baseline, which removes lexical context from our model, outperforms the full model. This is likely because the negative data is largely coherent at the trigram level (because it was generated from a trigram model), and the full model is much more sensitive to trigram coherence than the TREELET-RULE model. This also explains the poor performance of the 5-GRAM model.

We emphasize that the discriminative baselines are *specifically* trained to separate trigram text from natural English, while our model is trained on positive examples alone. Indeed, the methods in Post (2011) are simple binary classifiers, and it is not clear that these models would be properly calibrated for any other task, such as integration in a decoder.

One of the design goals of our system was that it be scalable. Unlike some of the discriminative baselines, which require expensive operations[9] on each training sentence, we can very easily scale our model to much larger amounts of data. In Table 4, we also show the performance of the generative models trained on our 1B corpus. All generative models improve, but TREELET-RULE remains the best, now outperforming the RERANK system, though of course it is likely that RERANK would improve if it could be scaled up to more training data.

[9]It is true that in order train our system, one must parse large amounts of training data, which can be costly, though it only needs to be done once. In contrast, even with observed training trees, the discriminative algorithms must still iteratively perform expensive operations (like parsing) for each sentence, and a new model must be trained for new types of negative data.

Model	**Pairwise**		**Independent**	
	WSJ	**1B**	**WSJ**	**1B**
PCFG	79.1	77.0	58.9	58.6
TREELET-RULE	90.3	94.4	63.8	**66.2**
TREELET	90.7	**94.5**	63.4	65.5
5-GRAM	86.3	93.5	55.7	60.1
HEADLEX	90.7	94.0	59.5	62.0
PCFG-LA	**91.3**	–	59.7	–
Foster et al. (2008)	–	–	**65.9**	–

Table 5: Classification accuracies on the noisy WSJ for models trained on WSJ Sections 2-21 and our 1B token corpus. "Pairwise" accuracy is the fraction of correct sentences whose SLR score was higher than its noisy version, and "independent" refers to standard binary classification accuracy.

5.3.2 "Noisy" Classification

We also evaluate the performance of our model on the task of distinguishing the noisy WSJ sentences of Foster et al. (2008) from their original versions. We use the noisy versions of Section 0 and 23 produced by their error-generating procedure. Because they only report classification results on Section 0, we used Section 23 to tune an SLR threshold, and tested our model on Section 0. We show the results of both independent and pairwise classification for the WSJ and 1B training sets in Table 5. Note that independent classification is much more difficult than for the trigram data, because sentences contain at most one change, which may not even result in an ungrammaticality. Again, our model outperforms the n-gram model for both types of classification, and achieves the same performance as the discriminative system of Foster et al. (2008), which is state-of-the-art for this data set. The TREELET-RULE system again slightly outperforms the full TREELET model at independent classification, though not at pairwise classification. This probably reflects the fact that semantic coherence can still influence the SLR score, despite our efforts to subtract it out. Because the TREELET model includes lexical context, it is more sensitive to seman-

	French	**German**	**Chinese**
5-GRAM	44.8	37.8	60.0
TREELET	**57.9**	**66.0**	**83.8**

Table 6: Pairwise comparison accuracy of MT output against a reference translation for French, German, and Chinese. The BLEU scores for these outputs are 32.7, 27.8, and 20.8. This task becomes easier, at least for our TREELET model, as translation quality drops. Cherry and Quirk (2008) report an accuracy of 71.9% on a similar experiment with German a source language, though the translation system and training data were different so the numbers are not comparable. In particular, their translations had a lower BLEU score, making their task easier.

tic coherence and thus more likely to misclassify semantically coherent but ungrammatical sentences. For pairwise comparisons, where semantic coherence is effectively held constant, such sentences are not problematic.

5.3.3 Machine Translation Classification

We follow Och et al. (2004) and Cherry and Quirk (2008) in evaluating our language models on their ability to distinguish the 1-best output of a machine translation system from a reference translation in a pairwise fashion. Unfortunately, we do not have access to the data used in those papers, so a direct comparison is not possible. Instead, we collected the English output of Moses (Hoang et al., 2007), using both French and German as source language, trained on the Europarl corpus used by WMT 2009.[10] We also collected the output of Joshua (Li et al., 2009) trained on 500K sentences of GALE Chinese-English parallel newswire. We trained both our TREELET model and a 5-GRAM model on the union of our 1B corpus and the English sides of our parallel corpora.

In Table 6, we show the pairwise comparison accuracy (using SLR) on these three corpora. We see that our system prefers the reference much more often than the 5-GRAM language model.[11] However, we also note that the easiness of the task is correlated with the quality of translations (as measured in BLEU score). This is not surprising – high-quality translations are often grammatical and even a perfect language model might not be able to differentiate such translations from their references.

[10]http://www.statmt.org/wmt09

[11]We note that the n-gram language model used by the MT system was much smaller than the 5-GRAM model, as they were only trained on the English sides of their parallel data.

5.4 Machine Translation Fluency

We also carried out reranking experiments on 1000-best lists from Moses using our syntactic language model as a feature. We did not find that the use of our syntactic language model made any statistically significant increases in BLEU score. However, we noticed in general that the translations favored by our model were more fluent, a useful improvement to which BLEU is often insensitive. To confirm this, we carried out an Amazon Mechanical Turk experiment where users from the United States were asked to compare translations using our TREELET language model as the language model feature to those using the 5-GRAM model.[12] We had 1000 such translation pairs rated by 4 separate Turkers each. Although these two hypothesis sets had the same BLEU score (up to statistical significance), the Turkers preferred the output obtained using our syntactic language model 59% of the time, indicating that our model had managed to pick out more fluent hypotheses that nonetheless were of the same BLEU score. This result was statistically significant with $p < 0.001$ using bootstrap resampling.

6 Conclusion

We have presented a simple syntactic language model that can be estimated using standard n-gram smoothing techniques on large amounts of data. Our model outperforms generative baselines on several evaluation metrics and achieves the same performance as state-of-the-art discriminative classifiers specifically trained on several types of negative data.

Acknowledgments

We would like to thank David Hall for some modeling suggestions and the anonymous reviewers for their comments. We thank both Matt Post and Jennifer Foster for providing us with their corpora. This work was partially supported by a Google Fellowship to the first author and by BBN under DARPA contract HR0011-12-C-0014.

[12]We used translations from the baseline Moses system of Section 5.3.3 with German as the input language. For each language model, we took k-best lists from the baseline system and replaced the baseline LM score with the new model's score. We then retrained all feature weights with MERT on the tune set, and selected the 1-best output on the test set.

References

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, Jeffrey Dean, and Google Inc. 2007. Large language models in machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the Association for Computational Linguistics*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the North American chapter of the Association for Computational Linguistics*.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the Association for Computational Linguistics*.

Ciprian Chelba. 1997. A structured language model. In *Proceedings of the Association for Computational Linguistics*.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. In *Proceedings of the Association for Computational Linguistics*.

Colin Cherry and Chris Quirk. 2008. Discriminative, syntactic language modeling through latent SVMs. In *Proceedings of The Association for Machine Translation in the Americas*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of Association for Computational Linguistics*.

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

Jennifer Foster, Joachim Wagner, and Josef van Genabith. 2008. Adapting a wsj-trained parser to grammatically noisy text. In *Proceedings of the Association for Computational Linguistics: Short Paper Track*.

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 *The Annual Conference of the Association for Computational Linguistics (ACL)*.

David Graff. 2003. English gigaword, version 3. In *Linguistic Data Consortium, Philadelphia, Catalog Number LDC2003T05*.

Keith Hall. 2004. *Best-first Word-lattice Parsing: Techniques for Integrated Syntactic Language Modeling*. Ph.D. thesis, Brown University.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

Hieu Hoang, Alexandra Birch, Chris Callison-burch, Richard Zens, Rwth Aachen, Alexandra Constantin, Marcello Federico, Nicola Bertoldi, Chris Dyer, Brooke Cowan, Wade Shen, Christine Moran, and Ondej Bojar. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Association for Computational Linguistics: Demonstration Session,*.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24.

Dan Klein and Chris Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of The Association for Machine Translation in the Americas*.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: an open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics*.

Franz J. Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A Smorgasbord of Features for Statistical Machine Translation. In *Proceedings of the North American Association for Computational Linguistic*.

Daisuke Okanohara and Jun'ichi Tsujii. 2007. A discriminative language model with pseudo-negative samples. In *Proceedings of the Association for Computational Linguistics*.

Adam Pauls and Dan Klein. 2011. Faster and smaller n-gram language models. In *Proceedings of the Association for Computational Linguistics*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL 2006*.

Matt Post and Daniel Gildea. 2009. Language modeling with tree substitution grammars. In *Proceedings* *of the Conference on Neural Information Processing Systems*.

Matt Post. 2011. Judging grammaticality with tree substitution grammar. In *Proceedings of the Association for Computational Linguistics: Short Paper Track*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of the Association of Computational Linguistics*.

Brian Roark. 2004. Probabilistic top-down parsing and language modeling. *Computational Linguistics*.

Ming Tan, Wenli Zhou, Lei Zheng, and Shaojun Wang. 2011. A large scale distributed syntactic, semantic and lexical language model for machine translation. In *Proceedings of the Association for Computational Linguistics*.

Ashish Vaswani, Haitao Mi, Liang Huang, and David Chiang. 2011. Rule markov models for fast tree-to-string translation. In *Proceedings of the Association for Computations Linguistics*.

Peng Xu, Ciprian Chelba, and Fred Jelinek. 2002. A study on richer syntactic dependencies for structured language modeling. In *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ying Zhang. 2009. *Structured language models for statistical machine translation*. Ph.D. thesis, Johns Hopkins University.