Less Grammar, More Features

David Hall **Greg Durrett** **Dan Klein**
Computer Science Division
University of California, Berkeley
{dlwh,gdurrett,klein}@cs.berkeley.edu

Abstract

We present a parser that relies primarily on extracting information directly from surface spans rather than on propagating information through enriched grammar structure. For example, instead of creating separate grammar symbols to mark the definiteness of an NP, our parser might instead capture the same information from the first word of the NP. Moving context out of the grammar and onto surface features can greatly simplify the structural component of the parser: because so many deep syntactic cues have surface reflexes, our system can still parse accurately with context-free backbones as minimal as X-bar grammars. Keeping the structural backbone simple and moving features to the surface also allows easy adaptation to new languages and even to new tasks. On the SPMRL 2013 multilingual constituency parsing shared task (Seddah et al., 2013), our system outperforms the top single parser system of Björkelund et al. (2013) on a range of languages. In addition, despite being designed for syntactic analysis, our system also achieves state-of-the-art numbers on the structural sentiment task of Socher et al. (2013). Finally, we show that, in both syntactic parsing and sentiment analysis, many broad linguistic trends can be captured via surface features.

1 Introduction

Naïve context-free grammars, such as those embodied by standard treebank annotations, do not parse well because their symbols have too little context to constrain their syntactic behavior. For example, *to* PPs usually attach to verbs and *of* PPs usually attach to nouns, but a context-free PP symbol can equally well attach to either. Much of the last few decades of parsing research has therefore focused on propagating contextual information from the leaves of the tree to internal nodes. For example, head lexicalization (Eisner, 1996; Collins, 1997; Charniak, 1997), structural annotation (Johnson, 1998; Klein and Manning, 2003), and state-splitting (Matsuzaki et al., 2005; Petrov et al., 2006) are all designed to take coarse symbols like PP and decorate them with additional context. The underlying reason that such propagation is even needed is that PCFG parsers score trees based on local configurations only, and any information that is not threaded through the tree becomes inaccessible to the scoring function. There have been non-local approaches as well, such as tree-substitution parsers (Bod, 1993; Sima'an, 2000), neural net parsers (Henderson, 2003), and rerankers (Collins and Koo, 2005; Charniak and Johnson, 2005; Huang, 2008). These non-local approaches can actually go even further in enriching the grammar's structural complexity by coupling larger domains in various ways, though their non-locality generally complicates inference.

In this work, we instead try to *minimize* the structural complexity of the grammar by moving as much context as possible onto local surface features. We examine the position that grammars should not propagate any information that is available from surface strings, since a discriminative parser can access that information directly. We therefore begin with a minimal grammar and iteratively augment it with rich input features that do not enrich the context-free backbone. Previous work has also used surface features in their parsers, but the focus has been on machine learning methods (Taskar et al., 2004), latent annotations (Petrov and Klein, 2008a; Petrov and Klein, 2008b), or implementation (Finkel et al., 2008).

By contrast, we investigate the extent to which

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 228–237,
Baltimore, Maryland, USA, June 23-25 2014.

we need a grammar at all. As a thought experiment, consider a parser with no grammar, which functions by independently classifying each span (i, j) of a sentence as an NP, VP, and so on, or *null* if that span is a non-constituent. For example, spans that begin with *the* might tend to be NPs, while spans that end with *of* might tend to be non-constituents. An independent classification approach is actually very viable for part-of-speech tagging (Toutanova et al., 2003), but is problematic for parsing – if nothing else, parsing comes with a structural requirement that the output be a well-formed, nested tree. Our parser uses a minimal PCFG backbone grammar to ensure a basic level of structural well-formedness, but relies mostly on features of surface spans to drive accuracy. Formally, our model is a CRF where the features factor over anchored rules of a small backbone grammar, as shown in Figure 1.

Some aspects of the parsing problem, such as the tree constraint, are clearly best captured by a PCFG. Others, such as heaviness effects, are naturally captured using surface information. The open question is whether surface features are adequate for key effects like subcategorization, which have deep definitions but regular surface reflexes (e.g. the preposition selected by a verb will often linearly follow it). Empirically, the answer seems to be yes, and our system produces strong results, e.g. up to 90.5 F1 on English parsing. Our parser is also able to generalize well across languages with little tuning: it achieves state-of-the-art results on multilingual parsing, scoring higher than the best single-parser system from the SPMRL 2013 Shared Task on a range of languages, as well as on the competition's average F1 metric.

One advantage of a system that relies on surface features and a simple grammar is that it is portable not only across languages but also across tasks to an extent. For example, Socher et al. (2013) demonstrates that sentiment analysis, which is usually approached as a flat classification task, can be viewed as tree-structured. In their work, they propagate real-valued vectors up a tree using neural tensor nets and see gains from their recursive approach. Our parser can be easily adapted to this task by replacing the X-bar grammar over treebank symbols with a grammar over the sentiment values to encode the output variables and then adding n-gram indicators to our feature set to capture the bulk of the lexical effects. When applied to this task, our system generally matches their accuracy overall and is able to outperform it on the overall sentence-level subtask.

2 Parsing Model

In order to exploit non-independent surface features of the input, we use a discriminative formulation. Our model is a conditional random field (Lafferty et al., 2001) over trees, in the same vein as Finkel et al. (2008) and Petrov and Klein (2008a). Formally, we define the probability of a tree T conditioned on a sentence $\mathbf{w}$ as

$$p(T|\mathbf{w}) \propto \exp\left(\theta^{\mathsf{T}} \sum_{r \in T} f(r, \mathbf{w})\right) \qquad (1)$$

where the feature domains r range over the (anchored) rules used in the tree. An anchored rule r is the conjunction of an unanchored grammar rule $\mathrm{rule}(r)$ and the start, stop, and split indexes where that rule is anchored, which we refer to as $\mathrm{span}(r)$. It is important to note that the richness of the backbone grammar is reflected in the structure of the trees T, while the features that condition directly on the input enter the equation through the anchoring $\mathrm{span}(r)$. To optimize model parameters, we use the Adagrad algorithm of Duchi et al. (2010) with L2 regularization.

We start with a simple X-bar grammar whose only symbols are NP, NP-bar, VP, and so on. Our base model has no surface features: formally, on each anchored rule r we have only an indicator of the (unanchored) rule identity, $\mathrm{rule}(r)$. Because the X-bar grammar is so minimal, this grammar does not parse very accurately, scoring just 73 F1 on the standard English Penn Treebank task.

In past work that has used tree-structured CRFs in this way, increased accuracy partially came from decorating trees T with additional annotations, giving a tree T' over a more complex symbol set. These annotations introduce additional context into the model, usually capturing linguistic intuition about the factors that influence grammaticality. For instance, we might annotate every constituent X in the tree with its parent Y, giving a tree with symbols $X[\hat{}Y]$. Finkel et al. (2008) used parent annotation, head tag annotation, and horizontal sibling annotation together in a single large grammar. In Petrov and Klein (2008a) and Petrov and Klein (2008b), these annotations were latent; they were inferred automatically during training.

Hall and Klein (2012) employed both kinds of annotations, along with lexicalized head word annotation. All of these past CRF parsers do also exploit span features, as did the structured margin parser of Taskar et al. (2004); the current work primarily differs in shifting the work from the grammar to the surface features.

The problem with rich annotations is that they increase the state space of the grammar substantially. For example, adding parent annotation can square the number of symbols, and each subsequent annotation causes a multiplicative increase in the size of the state space. Hall and Klein (2012) attempted to reduce this state space by factoring these annotations into individual components. Their approach changed the multiplicative penalty of annotation into an additive penalty, but even so their individual grammar projections are much larger than the base X-bar grammar.

In this work, we want to see how much of the expressive capability of annotations can be captured using surface evidence, with little or no annotation of the underlying grammar. To that end, we avoid annotating our trees at all, opting instead to see how far simple surface features will go in achieving a high-performance parser. We will return to the question of annotation in Section 5.

3 Surface Feature Framework

To improve the performance of our X-bar grammar, we will add a number of surface feature templates derived only from the words in the sentence. We say that an indicator is a surface property if it can be extracted without reference to the parse tree. These features can be implemented without reference to structured linguistic notions like headedness; however, we will argue that they still capture a wide range of linguistic phenomena in a data-driven way.

Throughout this and the following section, we will draw on motivating examples from the English Penn Treebank, though similar examples could be equally argued for other languages. For performance on other languages, see Section 6.

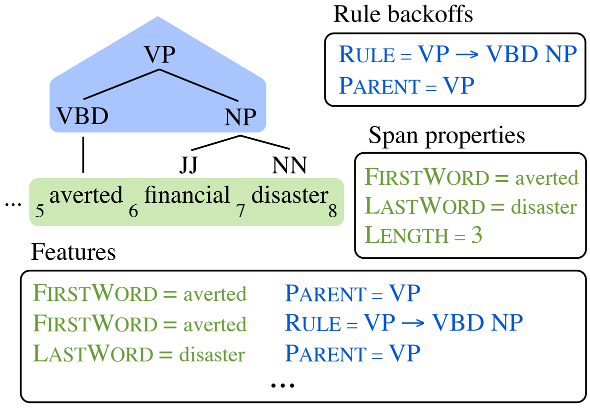

Figure 1: Features computed over the application of the rule VP → VBD NP over the anchored span *averted financial disaster* with the shown indices. Span properties are generated as described throughout Section 4; they are then conjoined with the rule and just the parent nonterminal to give the features fired over the anchored production.

Recall that our CRF factors over anchored rules r, where each r has identity $\mathrm{rule}(r)$ and anchoring $\mathrm{span}(r)$. The X-bar grammar has only indicators of $\mathrm{rule}(r)$, ignoring the anchoring. Let a *surface property* of r be an indicator function of $\mathrm{span}(r)$ and the sentence itself. For example, the first word in a constituent is a surface property, as is the word directly preceding the constituent. As illustrated in Figure 1, the actual features of the model are obtained by conjoining surface properties with various abstractions of the rule identity. For rule abstractions, we use two templates: the parent of the rule and the identity of the rule. The surface features are somewhat more involved, and so we introduce them incrementally.

One immediate computational and statistical issue arises from the sheer number of possible surface features. There are a great number of spans in a typical treebank; extracting features for every possible combination of span and rule is prohibitive. One simple solution is to only extract features for rule/span pairs that are actually observed in gold annotated examples during training. Because these "positive" features correspond to observed constituents, they are far less numerous than the set of all possible features extracted from all spans. As far as we can tell, all past CRF parsers have used "positive" features only.

However, negative features—features that are not observed in any tree—are still powerful indicators of (un)grammaticality: if we have never seen a PRN that starts with "has," or a span that begins with a quotation mark and ends with a close bracket, then we would like the model to be able to place negative weights on these features. Thus, we use a simple feature hashing scheme where positive features are indexed individually, while nega-

Features	Section	F1
RULE	4	73.0
+ SPAN FIRST WORD + SPAN LAST WORD + LENGTH	4.1	85.0
+ WORD BEFORE SPAN + WORD AFTER SPAN	4.2	89.0
+ WORD BEFORE SPLIT + WORD AFTER SPLIT	4.3	89.7
+ SPAN SHAPE	4.4	89.9

Table 1: Results for the Penn Treebank development set, reported in F1 on sentences of length ≤ 40 on Section 22, for a number of incrementally growing feature sets. We show that each feature type presented in Section 4 adds benefit over the previous, and in combination they produce a reasonably good yet simple parser.

tive features are bucketed together. During training there are no collisions between positive features, which generally receive positive weight, and negative features, which generally receive negative weight; only negative features can collide. Early experiments indicated that using a number of negative buckets equal to the number of positive features was effective.

4 Features

Our goal is to use surface features to replicate the functionality of other annotations, without increasing the state space of our grammar, meaning that the rules $\mathrm{rule}(r)$ remain simple, as does the state space used during inference.

Before we present our main features, we briefly discuss the issue of feature sparsity. While lexical features are a powerful driver of our parser, firing features on rare words would allow it to overfit the training data quite heavily. To that end, for the purposes of computing our features, a word is represented by its longest suffix that occurs 100 or more times in the training data (which will be the entire word, for common words).[1]

Table 1 shows the results of incrementally building up our feature set on the Penn Treebank development set. RULE specifies that we use only indicators on rule identity for binary production and nonterminal unaries. For this experiment and all others, we include a basic set of lexicon features, i.e. features on preterminal part-of-speech tags. A given preterminal unary at position i in the sentence includes features on the words (suffixes) at position $i-1$, i, and $i+1$. Because the lexicon is especially sensitive to morphological effects, we also fire features on all prefixes and suffixes of the current word up to length 5, regardless of frequency.

Subsequent lines in Table 1 indicate additional surface feature templates computed over the span, which are then conjoined with the rule identity as shown in Figure 1 to give additional features. In the rest of the section, we describe the features of this type that we use. Note that many of these features have been used before (Taskar et al., 2004; Finkel et al., 2008; Petrov and Klein, 2008b); our goal here is not to amass as many feature templates as possible, but rather to examine the extent to which a simple set of features can replace a complicated state space.

4.1 Basic Span Features

We start with some of the most obvious properties available to us, namely, the identity of the first and last words of a span. Because heads of constituents are often at the beginning or the end of a span, these feature templates can (noisily) capture monolexical properties of heads without having to incur the inferential cost of lexicalized annotations. For example, in English, the syntactic head of a verb phrase is typically at the beginning of the span, while the head of a simple noun phrase is the last word. Other languages, like Korean or Japanese, are more consistently head final.

Structural contexts like those captured by parent annotation (Johnson, 1998) are more subtle. Parent annotation can capture, for instance, the difference in distribution in NPs that have S as a parent (that is, subjects) and NPs under VPs (objects). We try to capture some of this same intuition by introducing a feature on the length of a span. For instance, VPs embedded in NPs tend to be short, usually as embedded gerund phrases. Because constituents in the treebank can be quite long, we bin our length features into 8 buckets, of

[1] Experiments with the Brown clusters (Brown et al., 1992) provided by Turian et al. (2010) in lieu of suffixes were not promising. Moreover, lowering this threshold did not improve performance.

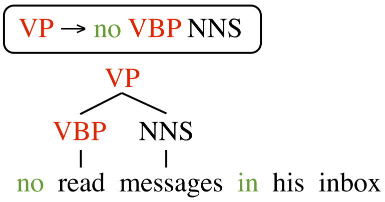

Figure 2: An example showing the utility of span context. The ambiguity about whether *read* is an adjective or a verb is resolved when we construct a VP and notice that the word proceeding it is unlikely.

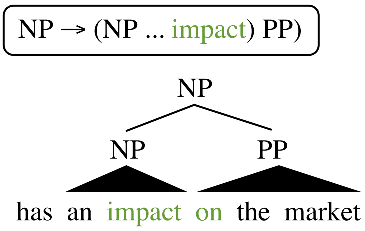

Figure 3: An example showing split point features disambiguating a PP attachment. Because *impact* is likely to take a PP, the monolexical indicator feature that conjoins *impact* with the appropriate rule will help us parse this example correctly.

lengths 1, 2, 3, 4, 5, 10, 20, and ≥21 words.

Adding these simple features (first word, last word, and lengths) as span features of the X-bar grammar already gives us a substantial improvement over our baseline system, improving the parser's performance from 73.0 F1 to 85.0 F1 (see Table 1).

4.2 Span Context Features

Of course, there is no reason why we should confine ourselves to just the words within the span: words outside the span also provide a rich source of context. As an example, consider disambiguating the POS tag of the word *read* in Figure 2. A VP is most frequently preceded by a subject NP, whose rightmost word is often its head. Therefore, we fire features that (separately) look at the words immediately preceding and immediately following the span.

4.3 Split Point Features

Another important source of features are the words at and around the split point of a binary rule application. Figure 3 shows an example of one instance of this feature template. *impact* is a noun that is more likely to take a PP than other nouns, and so we expect this feature to have high weight and encourage the attachment; this feature proves generally useful in resolving such cases of right-attachments to noun phrases, since the last word of the noun phrase is often the head. As another example, coordination can be represented by an indicator of the conjunction, which comes immediately after the split point. Finally, control structures with infinitival complements can be captured with a rule S → NP VP with the word "to" at the split point.

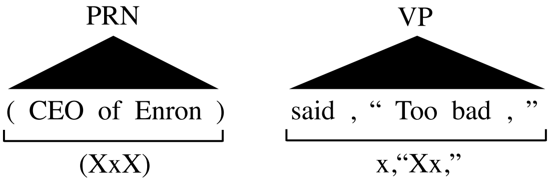

Figure 4: Computation of span shape features on two examples. Parentheticals, quotes, and other punctuation-heavy, short constituents benefit from being explicitly modeled by a descriptor like this.

4.4 Span Shape Features

We add one final feature characterizing the span, which we call span shape. Figure 4 shows how this feature is computed. For each word in the span,[2] we indicate whether that word begins with a capital letter, lowercase letter, digit, or punctuation mark. If it begins with punctuation, we indicate the punctuation mark explicitly. Figure 4 shows that this is especially useful in characterizing constructions such as parentheticals and quoted expressions. Because this feature indicates capitalization, it can also capture properties of NP internal structure relevant to named entities, and its sensitivity to capitalization and punctuation makes it useful for recognizing appositive constructions.

5 Annotations

We have built up a strong set of features by this point, but have not yet answered the question of whether or not grammar annotation is useful on top of them. In this section, we examine two of the most commonly used types of additional annotation, structural annotation, and lexical annotation.

[2] For longer spans, we only use words sufficiently close to the span's beginning and end.

Annotation	Dev, len ≤ 40
$v = 0, h = 0$	90.1
$v = 1, h = 0$	90.5
$v = 0, h = 1$	90.2
$v = 1, h = 1$	90.9
Lexicalized	90.3

Table 2: Results for the Penn Treebank development set, sentences of length ≤ 40, for different annotation schemes implemented on top of the X-bar grammar.

Recall from Section 3 that every span feature is conjoined with indicators over rules and rule parents to produce features over anchored rule productions; when we consider adding an annotation layer to the grammar, what that does is refine the rule indicators that are conjoined with every span feature. While this is a powerful way of refining features, we show that common successful annotation schemes provide at best modest benefit on top of the base parser.

5.1 Structural Annotation

The most basic, well-understood kind of annotation on top of an X-bar grammar is structural annotation, which annotates each nonterminal with properties of its environment (Johnson, 1998; Klein and Manning, 2003). This includes vertical annotation (parent, grandparent, etc.) as well as horizontal annotation (only partially Markovizing rules as opposed to using an X-bar grammar).

Table 2 shows the performance of our feature set in grammars with several different levels of structural annotation.[3] Klein and Manning (2003) find large gains (6% absolute improvement, 20% relative improvement) going from $v = 0, h = 0$ to $v = 1, h = 1$; however, we do not find the same level of benefit. To the extent that our parser needs to make use of extra information in order to apply a rule correctly, simply inspecting the input to determine this information appears to be almost as effective as relying on information threaded through the parser.

In Section 6 and Section 7, we use $v = 1$ and $h = 0$; we find that $v = 1$ provides a small, reliable improvement across a range of languages and tasks, whereas other annotations are less clearly beneficial.

	Test ≤ 40	Test all
Berkeley	90.6	90.1
This work	89.9	89.2

Table 3: Final Parseval results for the $v = 1, h = 0$ parser on Section 23 of the Penn Treebank.

5.2 Lexical Annotation

Another commonly-used kind of structural annotation is lexicalization (Eisner, 1996; Collins, 1997; Charniak, 1997). By annotating grammar nonterminals with their headwords, the idea is to better model phenomena that depend heavily on the semantics of the words involved, such as coordination and PP attachment.

Table 2 shows results from lexicalizing the X-bar grammar; it provides meager improvements. One probable reason for this is that our parser already includes monolexical features that inspect the first and last words of each span, which captures the syntactic or the semantic head in many cases or can otherwise provide information about what the constituent's type may be and how it is likely to combine. Lexicalization allows us to capture bilexical relationships along dependency arcs, but it has been previously shown that these add only marginal benefit to Collins's model anyway (Gildea, 2001).

5.3 English Evaluation

Finally, Table 3 shows our final evaluation on Section 23 of the Penn Treebank. We use the $v = 1, h = 0$ grammar. While we do not do as well as the Berkeley parser, we will see in Section 6 that our parser does a substantially better job of generalizing to other languages.

6 Other Languages

Historically, many annotation schemes for parsers have required language-specific engineering: for example, lexicalized parsers require a set of head rules and manually-annotated grammars require detailed analysis of the treebank itself (Klein and Manning, 2003). A key strength of a parser that does not rely heavily on an annotated grammar is that it may be more portable to other languages. We show that this is indeed the case: on nine languages, our system is competitive with or better than the Berkeley parser, which is the best single

[3] We use $v = 0$ to indicate no annotation, diverging from the notation in Klein and Manning (2003).

	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Avg
	Dev, all lengths									
Berkeley	78.24	69.17	79.74	81.74	87.83	83.90	70.97	84.11	74.50	78.91
Berkeley-Rep	78.70	84.33	79.68	82.74	89.55	89.08	82.84	87.12	75.52	83.28
Our work	78.89	83.74	79.40	83.28	88.06	87.44	81.85	91.10	75.95	83.30
	Test, all lengths									
Berkeley	79.19	70.50	80.38	78.30	86.96	81.62	71.42	79.23	79.18	78.53
Berkeley-Tags	78.66	74.74	79.76	78.28	85.42	85.22	78.56	86.75	80.64	80.89
Our work	78.75	83.39	79.70	78.43	87.18	88.25	80.18	90.66	82.00	83.17

Table 4: Results for the nine treebanks in the SPMRL 2013 Shared Task; all values are F-scores for sentences of all lengths using the version of `evalb` distributed with the shared task. Berkeley-Rep is the best single parser from (Björkelund et al., 2013); we only compare to this parser on the development set because neither the system nor test set values are publicly available. Berkeley-Tags is a version of the Berkeley parser run by the task organizers where tags are provided to the model, and is the best single parser submitted to the official task. In both cases, we match or outperform the baseline parsers in aggregate and on the majority of individual languages.

parser[4] for the majority of cases we consider.

We evaluate on the constituency treebanks from the Statistical Parsing of Morphologically Rich Languages Shared Task (Seddah et al., 2013). We compare to the Berkeley parser (Petrov and Klein, 2007) as well as two variants. First, we use the "Replaced" system of Björkelund et al. (2013) (Berkeley-Rep), which is their best single parser.[5] The "Replaced" system modifies the Berkeley parser by replacing rare words with morphological descriptors of those words computed using language-specific modules, which have been hand-crafted for individual languages or are trained with additional annotation layers in the treebanks that we do not exploit. Unfortunately, Björkelund et al. (2013) only report results on the development set for the Berkeley-Rep model; however, the task organizers also use a version of the Berkeley parser provided with parts of speech from high-quality POS taggers for each language (Berkeley-Tags). These part-of-speech taggers often incorporate substantial knowledge of each language's morphology. Both Berkeley-Rep and Berkeley-Tags make up for some shortcomings of the Berkeley parser's unknown word model, which is tuned to English.

In Table 4, we see that our performance is overall substantially higher than that of the Berkeley parser. On the development set, we outperform the Berkeley parser and match the performance of the Berkeley-Rep parser. On the test set, we outperform both the Berkeley parser and the Berkeley-Tags parser on seven of nine languages, losing only on Arabic and French.

These results suggest that the Berkeley parser may be heavily fit to English, particularly in its lexicon. However, even when language-specific unknown word handling is added to the parser, our model still outperforms the Berkeley parser overall, showing that our model generalizes even better across languages than a parser for which this is touted as a strength (Petrov and Klein, 2007). Our span features appear to work well on both head-initial and head-final languages (see Basque and Korean in the table), and the fact that our parser performs well on such morphologically-rich languages as Hungarian indicates that our suffix model is sufficient to capture most of the morphological effects relevant to parsing. Of course, a language that was heavily prefixing would likely require this feature to be modified. Likewise, our parser does not perform as well on Arabic and Hebrew. These closely related languages use templatic morphology, for which suffixing is not appropriate; however, using additional surface features based on the output of a morphological analyzer did not lead to increased performance.

Finally, our high performance on languages such as Polish and Swedish, whose training treebanks consist of 6578 and 5000 sentences, respectively, show that our feature-rich model performs robustly even on treebanks much smaller than the Penn Treebank.[6]

[4] I.e. it does not use a reranking step or post-hoc combination of parser results.

[5] Their best parser, and the best overall parser from the shared task, is a reranked product of "Replaced" Berkeley parsers.

[6] The especially strong performance on Polish relative to other systems is partially a result of our model being able to produce unary chains of length two, which occur frequently in the Polish treebank (Björkelund et al., 2013).

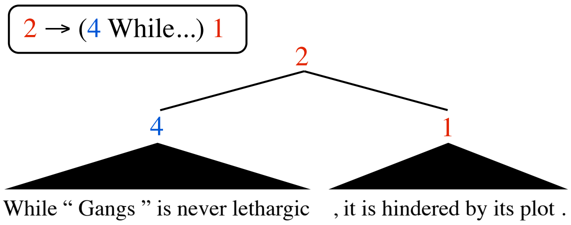
2 → (4 While...) 1
2
4
1
While " Gangs " is never lethargic
, it is hindered by its plot .

Figure 5: An example of a sentence from the Stanford Sentiment Treebank which shows the utility of our span features for this task. The presence of "While" under this kind of rule tells us that the sentiment of the constituent to the right dominates the sentiment to the left.

7 Sentiment Analysis

Finally, because the system is, at its core, a classifier of spans, it can be used equally well for tasks that do not normally use parsing algorithms. One example is sentiment analysis. While approaches to sentiment analysis often simply classify the sentence monolithically, treating it as a bag of n-grams (Pang et al., 2002; Pang and Lee, 2005; Wang and Manning, 2012), the recent dataset of Socher et al. (2013) imposes a layer of structure on the problem that we can exploit. They annotate every constituent in a number of training trees with an integer sentiment value from 1 (very negative) to 5 (very positive), opening the door for models such as ours to learn how syntax can structurally affect sentiment.[7]

Figure 5 shows an example that requires some analysis of sentence structure to correctly understand. The first constituent conveys positive sentiment with *never lethargic* and the second conveys negative sentiment with *hindered*, but to determine the overall sentiment of the sentence, we need to exploit the fact that *while* signals a discounting of the information that follows it. The grammar rule $2 \rightarrow 4\ 1$ already encodes the notion of the sentiment of the right child being dominant, so when this is conjoined with our span feature on the first word (*While*), we end up with a feature that captures this effect. Our features can also lexicalize on other discourse connectives such as *but* or *however*, which often occur at the split point between two spans.

7.1 Adapting to Sentiment

Our parser is almost entirely unchanged from the parser that we used for syntactic analysis. Though the treebank grammar is substantially different, with the nonterminals consisting of five integers with very different semantics from syntactic nonterminals, we still find that parent annotation is effective and otherwise additional annotation layers are not useful.

One structural difference between sentiment analysis and syntactic parsing lies in where the relevant information is present in a span. Syntax is often driven by heads of constituents, which tend to be located at the beginning or the end, whereas sentiment is more likely to depend on modifiers such as adjectives, which are typically present in the middle of spans. Therefore, we augment our existing model with standard sentiment analysis features that look at unigrams and bigrams in the span (Wang and Manning, 2012). Moreover, the Stanford Sentiment Treebank is unique in that each constituent was annotated in isolation, meaning that context never affects sentiment and that every word always has the same tag. We exploit this by adding an additional feature template similar to our span shape feature from Section 4.4 which uses the (deterministic) tag for each word as its descriptor.

7.2 Results

We evaluated our model on the fine-grained sentiment analysis task presented in Socher et al. (2013) and compare to their released system. The task is to predict the root sentiment label of each parse tree; however, because the data is annotated with sentiment at each span of each parse tree, we can also evaluate how well our model does at these intermediate computations. Following their experimental conditions, we filter the test set so that it only contains trees with non-neutral sentiment labels at the root.

Table 5 shows that our model outperforms the model of Socher et al. (2013)—both the published numbers and latest released version—on the task of root classification, even though the system was not explicitly designed for this task. Their model has high capacity to model complex interactions of words through a combinatory tensor, but it appears that our simpler, feature-driven model is just as effective at capturing the key effects of compositionality for sentiment analysis.

[7] Note that the tree structure is assumed to be given; the problem is one of labeling a fixed parse backbone.

	Root	All Spans
Non-neutral Dev (872 trees)		
Stanford CoreNLP current	50.7	80.8
This work	53.1	80.5
Non-neutral Test (1821 trees)		
Stanford CoreNLP current	49.1	80.2
Stanford EMNLP 2013	45.7	80.7
This work	49.6	80.4

Table 5: Fine-grained sentiment analysis results on the Stanford Sentiment Treebank of Socher et al. (2013). We compare against the printed numbers in Socher et al. (2013) as well as the performance of the corresponding release, namely the sentiment component in the latest version of the Stanford CoreNLP at the time of this writing. Our model handily outperforms the results from Socher et al. (2013) at root classification and edges out the performance of the latest version of the Stanford system. On all spans of the tree, our model has comparable accuracy to the others.

8 Conclusion

To date, the most successful constituency parsers have largely been generative, and operate by refining the grammar either manually or automatically so that relevant information is available locally to each parsing decision. Our main contribution is to show that there is an alternative to such annotation schemes: namely, conditioning on the input and firing features based on anchored spans. We build up a small set of feature templates as part of a discriminative constituency parser and outperform the Berkeley parser on a wide range of languages. Moreover, we show that our parser is adaptable to other tree-structured tasks such as sentiment analysis; we outperform the recent system of Socher et al. (2013) and obtain state of the art performance on their dataset.

Our system is available as open-source at `https://www.github.com/dlwh/epic`.

Acknowledgments

This work was partially supported by BBN under DARPA contract HR0011-12-C-0014, by a Google PhD fellowship to the first author, and an NSF fellowship to the second. We further gratefully acknowledge a hardware donation by NVIDIA Corporation.

References

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (Re)ranking Meets Morphosyntax: State-of-the-art Results from the SPMRL 2013 Shared Task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Rens Bod. 1993. Using an Annotated Corpus As a Stochastic Grammar. In *Proceedings of the Sixth Conference on European Chapter of the Association for Computational Linguistics*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine N-best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Eugene Charniak. 1997. Statistical Techniques for Natural Language Parsing. *AI Magazine*, 18:33–44.

Michael Collins and Terry Koo. 2005. Discriminative Reranking for Natural Language Parsing. *Computational Linguistics*, 31(1):25–70, March.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL*, pages 16–23.

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *COLT*.

Jason Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL 2008*, pages 959–967.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of Empirical Methods in Natural Language Processing*.

David Hall and Dan Klein. 2012. Training factored PCFGs with expectation propagation. In *EMNLP*.

James Henderson. 2003. Inducing History Representations for Broad Coverage Statistical Parsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June. Association for Computational Linguistics.

Mark Johnson. 1998. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24(4):613–632, December.

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

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*, pages 75–82, Morristown, NJ, USA.

Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs Up?: Sentiment Classification Using Machine Learning Techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *NAACL-HLT*.

Slav Petrov and Dan Klein. 2008a. Discriminative log-linear grammars with latent variables. In *NIPS*, pages 1153–1160.

Slav Petrov and Dan Klein. 2008b. Sparse multi-scale grammars for discriminative latent variable parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 867–876, Honolulu, Hawaii, October. Association for Computational Linguistics.

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.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, and Alina Wróblewska. 2013. Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Khalil Sima'an. 2000. Tree-gram Parsing Lexical Dependencies and Structural Relations. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of Empirical Methods in Natural Language Processing*.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning. 2004. Max-Margin Parsing. In *In Proceedings of Empirical Methods in Natural Language Processing*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Sida Wang and Christopher Manning. 2012. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.