Grammarless Parsing for Joint Inference

Jason Naradowsky[1,2] *Tim Vieira*[3] *David A. Smith*[4]
(1) University of Massachusetts Amherst, Amherst, Massachusetts
(2) Macquarie University, Sydney, Australia
(3) Johns Hopkins University, Baltimore, Maryland
(4) Northeastern University, Boston, Massachusetts
narad@cs.umass.edu, tim@cs.jhu.edu, dasmith@ccs.neu.edu

Abstract

Many NLP tasks interact with syntax. The presence of a named entity span, for example, is often a clear indicator of a noun phrase in the parse tree, while a span in the syntax can help indicate the lack of a named entity in the spans that cross it. For these types of problems joint inference offers a better solution than a pipelined approach, and yet large joint models are rarely pursued. In this paper we argue this is due in part to the absence of a general framework for joint inference which can efficiently represent syntactic structure.

We propose an alternative and novel method in which constituency parse constraints are imposed on the model via combinatorial factors in a Markov random field, guaranteeing that a variable configuration forms a valid tree. We apply this approach to jointly predicting parse and named entity structure, for which we introduce a zero-order semi-CRF named entity recognizer which also relies on a combinatorial factor. At the junction between these two models, soft constraints coordinate between syntactic constituents and named entity spans, providing an additional layer of flexibility on how these models interact. With this architecture we achieve the best-reported results on both CRF-based parsing and named entity recognition on sections of the OntoNotes corpus, and outperform state-of-the-art parsers on an NP-identification task, while remaining asymptotically faster than traditional grammar-based parsers.

1 Introduction

Research in statistical parsing has made significant progress on recovering the kinds of annotations found in treebank-annotated corpora (Collins, 2003; Petrov et al., 2006), but in practice parsing is rarely an end in itself. Instead, parsing must be integrated with other forms of markup (part-of-speech tagging, named-entity recognition, coreference resolution) in order to perform complete end-to-end tasks like question answering. Historically, independent pursuit of these tasks has often been accompanied by the belief that improvements on a measure intrinsic to parsing (for our domain, often F1 on the test section of the Wall Street Journal) will accurately reflect improvements on an end task when incorporated into a complete system.

Nevertheless, errors propagate in NLP pipelines, and the need for a consistent analysis across several layers of linguistic structure has motivated the development of joint models in which uncertainty is shared between individual model components. Previous work has applied joint inference to parsing and named-entity recognition (NER), the task pursued in this paper, by augmenting the grammar with specialized non-terminals which couple syntactic and NER labels to represent an analysis over both domains (Finkel and Manning, 2009). This coupling proved to be beneficial to both tasks, increasing parsing performance by 0.47 F1 and NER performance by 3.27 F1 on average over the independently-trained models.

Yet there are many reasons one might be concerned with this approach. First, it is a problem-specific formulation of a joint problem, and not necessarily extensible to problems that require a looser or more flexible coupling between the two problem domains. For instance, the presence of an NER span indicates the presence of a noun phrase in the corresponding syntax. NER spans are therefore a subset of syntactic constituent spans, and the two problems can be represented with a single tree-structured derivation. However, a joint model of prosody and parsing would be difficult to capture with this approach, as boundaries of prosodic and syntactic spans are not required to overlap (Selkirk, 1984; Jackendoff, 2002, Ch. 5, p. 118-123).

Second, formulating joint inference as context-free parsing also imposes a computational burden at a particularly susceptible point in the algorithm. Consider the CKY algorithm's complexity of $O(|G|n^3)$, where $|G|$ is the size of the grammar and n the length of the sentence. While algorithms are generally compared in terms of their asymptotic complexity (here the property of being a cubic-time algorithm), in many natural language problems the grammar constant is by far the most computationally expensive component. In the grammar augmentation approach to joint inference, the grammar constant increases as the product of the different domains' label sets.

Perhaps the most limiting aspect of this approach is that it makes assumptions over what types of structures are permissible, which may force the model structure of one problem into less intuitive designs based on the structure of another. For instance, named entity recognizers have traditionally been developed using sequence models, but some of the sequential model structure is lost when conforming to a grammar-based notion of joint inference.[1]

All of the aforementioned criticisms will only become more pertinent when joint inference in NLP begins to scale toward coupling the three or more problems in larger end-to-end systems. We therefore present in this paper a more generalized approach to joint inference via combinatorial factor constraints in factor graphs. In particular we describe a special-purpose combinatorial factor for constraining syntactic variables to a valid constituent bracketing, providing an efficient and flexible method for representing constituent syntax in graphical models. We first demonstrate that on its own this simple parser setup is competitive in accuracy with more complex models at the standard task of recovering treebank annotation. Then, we demonstrate the composability of the factor graph approach by coupling the syntactic representation to a semi-Markov NER model and performing inference jointly.

2 Parsing without a Grammar

While the dominant approaches to constituency parsing almost always depend on an explicit grammar, efficiently representing grammatical parsing in frameworks as general as factor graphs is very difficult. In this section we will discuss methods for representing constituency-style parsing constraints in a factor graph. This general-purpose syntactic representation can then be flexibly coupled to other tasks, and trained based upon common inference and learning methods.

A factor graph is a type of graphical model represented as a bipartite graph composed of variables, whose values represent the state of the world, and factors, which specify potentials over the belief of a particular state. For learning methods that require the marginal beliefs to compute a gradient, factor graphs provide an efficient way of computing such marginals via message passing inference algorithms.

[1] In this particular case, one could compose the context-free grammar with the finite-state model—causing the grammar to grow even more. Similar problems arise in combining syntactic machine translation models with n-gram language models (Vaswani et al., 2011).

The difficulty in representing grammars with factor graphs is that the complexity of inference in such a graph scales exponentially in the size of the largest clique. Naively representing a weighted grammar in Chomsky normal form requires variables to be densely connected with ternary factors [2] to represent the application of a weighted grammar rule. This is not only computationally problematic (on a per-iteration of inference basis), but could also further complicate convergence when performing belief propagation, an efficient message passing inference algorithm (Pearl, 1988), in the now exceedingly loopy graph.

If we were to back off from a desire to represent grammatical parsing in a factor graph, we could impose a constraint which prohibits crossing brackets by examining variables in a pairwise manner and assigning zero probability to configurations in which two conflicting variables were both on, and this would be possible with a quartic number of constraining factors. But while both approaches would capture the gist of a useful constraint, they are both computationally inefficient and will still fail to prohibit all structures that are not valid trees.

Instead we introduce a special-purpose combinatorial factor, CKYTree. It was observed previously (Smith and Eisner, 2008) that the outgoing messages from such combinatorial factors to a variable could be computed from the factor's posterior beliefs about that variable, thus defining an interface for inserting special purpose logic within the standard inference algorithm. Instead of representing a tree constraint externally in the structure of the model, we can encapsulate similar logic in this factor where the computation can be done more efficiently with variants of standard dynamic programming algorithms (Figure 1). Previous work has applied this technique to non-projective and projective dependency parsing (Smith and Eisner, 2008; Naradowsky et al., 2012), and a similar belief propagation approach has been used to enforce well-formed productions in CCG parsing (Auli and Lopez, 2011).

More specifically, inputs to this algorithm are the span weights $u(i, j)$. As in earlier dependency parsing work, these weights are derived from the ratio of messages coming in from the $Span$ variables:

$$u(i,j) = \frac{m_{Span(i,j)\to\text{CKYTree}}(\text{true})}{m_{Span(i,j)\to\text{CKYTree}}(\text{false})}$$

After running this inside-outside algorithm in $O(n^3)$ time, we calculate the $O(n^2)$ outgoing messages from CKYTree:

$$\begin{aligned} m_{\text{CKYTree}\to Span(i,j)}(\text{true}) &= g(i,j) \\ m_{\text{CKYTree}\to Span(i,j)}(\text{false}) &= 1 - u(i,j)\cdot g(i,j) \end{aligned}$$

Here $g(i, j)$ is the gradient of the sum of the weights of all trees with respect to the input weight $u(i, j)$. Using the familiar inside β and outside α quantities, we can write this as:

$$g(i,j) \overset{\text{def}}{=} \frac{\partial \beta(0,n)}{\partial u(i,j)} = \frac{\alpha(i,j)}{\beta(0,n)}$$

[2] Ternary factors connect three variables, in this case a parent span variable and two contained and adjacent child span variables.

Algorithm 1 Bracket inside algorithm

```
function Inside($u, n$)
    for $w \leftarrow 2..n$ do
        for $i \leftarrow 0..(n-w)$ do
            $k \leftarrow i + w$
            $s \leftarrow \mathbf{0}$
            for $j \leftarrow (i+1)..(k-1)$ do
                $s \overset{\oplus}{\leftarrow} \beta(i,j) \otimes \beta(j,k)$
            end for
            $\beta(i,k) \leftarrow s \oplus u(i,k)$
        end for
    end for
    return $\beta$
end function
```

Algorithm 2 Bracket outside algorithm

```
function Outside($u, \beta, n$)
    for $w \leftarrow n..2$ do
        for $i \leftarrow 0..(n-w)$ do
            $k \leftarrow i + w$
            for $j \leftarrow (i+1)..(k-1)$ do
                $\alpha(i,j) \overset{\oplus}{\leftarrow} \alpha(i,k) \otimes \beta(j,k) \otimes u(i,k)$
                $\alpha(j,k) \overset{\oplus}{\leftarrow} \alpha(i,k) \otimes \beta(i,j) \otimes u(i,k)$
                $g(i,k) \overset{\oplus}{\leftarrow} \alpha(i,k) \otimes \beta(i,j) \otimes \beta(j,k)$
            end for
        end for
    end for
    return $g$
end function
```

Figure 1: Bracket inference algorithms are special cases of the familiar inside and outside algorithms for PCFGs (Baker, 1979) with a different "grammar-rule" weight $u(i,j)$ for each span.

2.1 Bracket Model

Having introduced the necessary computational framework, we can now present our most basic model of syntax: a factor graph for predicting unlabeled, projective binary constituency trees without any representation of a grammar. We model the possible parses of an n-word sentence with the following factor graph:[3]

- Let $\{Span(i,j) : 0 \leq i < j \leq n\}$ be $O(n^2)$ boolean variables such that $Span(i,j) = \text{true}$ iff there is a bracket spanning i to j.[4]
- Let $\{\text{Brack}(i,j) : 0 \leq i < j \leq n\}$ be $O(n^2)$ unary factors, each attached to the corresponding variable $Span(i,j)$. These factors score the independent suitability of each span to appear in an unlabeled constituency tree.
- Let CKYTree be a global combinatorial factor attached to all the $Span(i,j)$ variables. This factor contributes a factor of 1 to the model's score if the span variables collectively form a legal, binary bracketing and a factor of 0 otherwise. It enforces, therefore, a hard constraint on the variables. All outgoing messages from this factor are computed simultaneously by the outside algorithm described above in §2.

This comprises the core of our parsing model, which couples local span-oriented factors with rich features to a combinatorial factor that guarantees the global structure is a valid constituent tree. Though it is lightweight, with a complexity of $O(n^3 + n^2)$ in comparison to $O(|G|n^3)$ of a traditional CKY parser, it is not possible to identify which predicted spans are introduced through binarization (because constituents are not labeled), and thus limits the validity of parser comparison. Instead of evaluating this model in isolation, we immediately augment it with factors and variables to model *labeled* constituency trees.

[3]Variables are denoted by italicized names, factors by small capitals.

[4]In practice, we do not need to include variables for spans of width 1 or n, since they will always be true.

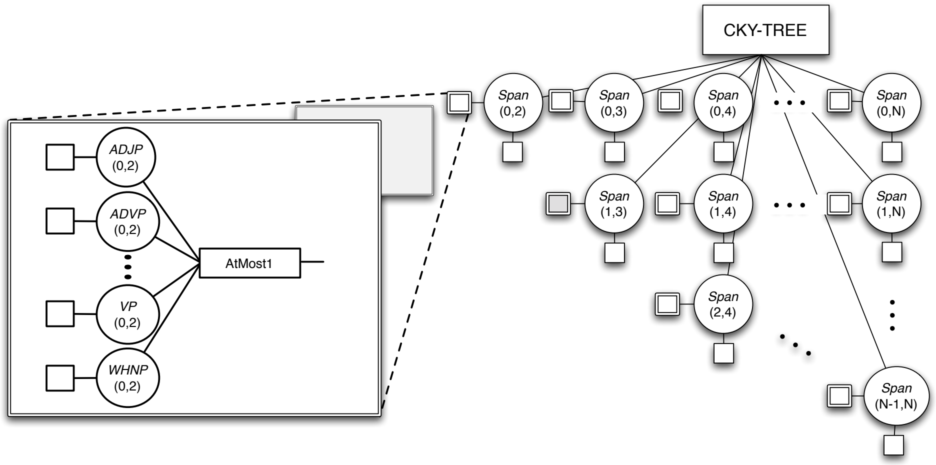

Figure 2: A graphical depiction of Bracket Model and Labeled Bracket Model. The components of the Bracket Model – *Span(i,j)* variables, corresponding Brack(i,j) factors (shown as single-bound rectangles), and the constraining tree factor – are represented on the right. The Labeled Bracket Model extensions include the AtMost1 factors on each span variable (shown as double-bound rectangles), which coordinate to select just one label from the set of nonterminals (illustrated left).

2.2 Labeled Bracket Model (LBM)

Extending the bracket model to incorporate a set of labels L is as simple as connecting each $Span(i,j)$ variable to a set of $Label$ variables via a factor that ensures at most one label is present:

- Let $\{Label(i,j,\lambda) : \lambda \in L$, and $0 \leq i < j \leq n\}$ be $O(|L|n^2)$ boolean variables such that $Label(i,j,\lambda) = \text{true}$ iff there is a bracket spanning i to j with constituent label λ.

- Let $\{\text{AtMost1}(i,j) : 0 \leq i < j \leq n\}$ be $O(n^2)$ factors, each coordinating the set L of possible nonterminal variables to the $Span$ variable at each i,j tuple, allowing a label variable to be true iff all other label variables are false and $Span(i,j) = \text{true}$.

One contribution of this paper is the exploitation of logical factors to proxy multinomial distributions. In this case the AtMost1 factors coordinate sets of label variables with a boolean indicator variable, capturing the same semantics as the multinomial while, most importantly, partitioning the values themselves into separate boolean variables. This is a necessary design choice for easily extending the model with grammatical rules as discussed in Section §6.

Asymptotically this is still an advantageous decomposition, yielding a complexity of $O(n^3 + |L|n^2)$, though the label set must also include a separate nonterminal for labels introduced through binarization of the original grammar, so predicted spans labeled with these binarized symbols can be removed when constructing the final parse tree. While there is some freedom in this choice and a traditional CNF transformation often introduces a binarized variant of each constituent, we find no appreciable benefit from having more than one binary symbol.

2.3 Inference

With the exception of combinatorial factors, which have their own specialized propagators as described above, inference is performed using the standard sum-product algorithm (Pearl, 1988), also known as *belief propagation* (BP). Each node in the graph sends and receives messages of two types:

A message from a variable node v to a factor node u is the product of all the messages from the factor nodes connected to v excluding u itself.

$$m_{v\to u}(x_v) = \prod_{u*\in N(v)/u} m_{u*\to v}(x_v)$$

Similarly, a message from a factor node u to a variable node v is the product of the factor and all of the messages from neighboring variables, marginalizing over all neighboring variables with the exception of x_v.

$$m_{u\to v}(x_v) = \sum_{x'_u : x'_v = x_v} f_u(X'_u) \prod_{v*\in N(u)/v} m_{v*\to u}(x'_{v*})$$

In cases where u is a combinatorial factor, its own combinatorial algorithm is executed instead of the standard update, but inference otherwise continues without change.

Both the bracket and label models form non-loopy graphs and, in a manner analogous to the forward-backward algorithm on chains, converge in two iterations of message passing. The robustness of this approach to joint inference with syntax is that regardless of the model structure that may be created for other tasks or couplings, the inference algorithm can remain the same. If loops arise in the graph then the convergence guarantee is lost, but in practice we find that loopy BP, which is fundamentally the same algorithm run on cyclic graphs, often converges within several iterations for joint parsing and NER tasks.

2.4 Unary Re-writes

While our motivation here is to present a syntactic representation which can be used to the benefit of a separate NLP end task, we also assess the performance of the parser independently, and must therefore address the rather tangential issue of unary rewrites. In a context-free grammar, internal unary productions—nonterminals with a single nonterminal child—fall out naturally from the recursive definition of the model, but in our span-factored formulation they must be handled separately. Similar work in conditional random field parsing (Finkel et al., 2008) collapses unary rewrite chains to single augmented rules while prohibiting multiple rewrites from occupying the same span, but this comes at the cost of additional complexity during decoding. Other work has attempted to separately predict leaf-level unary rewrites, albeit for the purpose of improving decoding speed (Bodenstab et al., 2011).

We follow the latter approach, pruning unary rewrites from the data entirely (replacing them with their parent constituent) and training a separate log-linear classifier for reinserting unary rewrites into the trees produced by the parser. Because the distribution of unary rewrites is so overwhelmingly concentrated in the nodes immediately governing the leaves of the tree, we focus our efforts solely on predicting these unary rewrites, accepting the performance hit of not predicting any which may appear in the body of the tree (comprising between 11% and 15% of all unary spans newswire sections of OntoNotes).

3 Named Entity Model

To demonstrate the usefulness of this syntactic representation in joint inference tasks, we choose to evaluate primarily on the end task of named entity extraction. In contrast to previous work, we avoid augmenting the grammar with special non-terminals and construct the model once again within the context of a factor graph. This choice allows us to emphasize the sequential information that historically has produced state-of-the-art performance.

As with parsing, however, incorporating some state-of-the-art models is not a trivial task. Consider for instance a semi-Markov conditional random field (semi-CRF) model (Sarawagi and Cohen, 2004). The context-rich nature of these models is very difficult to capture within the top-down derivations of a tree-based decoder for joint inference. Even in the general framework of a factor graph, representing these structures efficiently in a manner compatible with the rest of the joint architecture presents a challenge. However, we can once again encapsulate this logic within one combinatorial factor and connect it in a global fashion to all variables corresponding to NER spans. First, we assume a named entity variable set analogous to the LBM (denoted $NER\text{-}Span$ and $NER\text{-}Label$ and behaving similarly).

- Let Semi-CRF be a combinatorial constraint connected to all $NER\text{-}Span(i, j)$ variables. The factor implements a weighted Semi-CRF with a maximum span width μ, as described in (Sarawagi and Cohen, 2004), over the log-odds of each span variable's boolean values. Unlike the valid-bracketing constraint imposed by the CKYTree factor, this effect can be achieved with a polynomial number of binary factors. The $O(\mu^2 n^2)$ such factors would lead to inefficient inference in a very loopy factor graph. In all experiments $\mu = 10$.

4 Joint NER and Parsing

Perhaps the main advantage of casting constituency parsing in terms of a factor graph is the ease with which the model can be extended to improve separate but related task. To illustrate this, we couple our LBM constituency parser to the span-based model of named-entity recognition with an additional type of specialized factor:

- Let $\{\text{NER-Nand}(i, j) : 0 \leq i < j \leq n; 1 < j - i \leq \mu\}$ be a set of at most $O(n^2)$ factors coordinating syntactic $Span(i, j)$ and named entity span variable $NER\text{-}Span(i, j)$, multiplying in 1 unless both variables are on, in which case it multiplies a connective potential $\phi(i, j)$ derived from its features. Intuitively the joint model might learn features weights such that $\phi(i, j) > 1$, i.e., constituents and NER spans are more likely to be coterminous. The number of these coordinating factors is constrained to the number of NER span variables, subject to the maximum span-width $\mu = 10$.

These special purpose factors allow the model to learn how to best coordinate the sub-problems, adding an additional layer of flexibility to the joint architecture. For this particular domain, features that are useful for distinguishing noun phrase spans from other spans are good candidates, as essentially all named entities correspond to noun phrases in the syntax.

5 Experiments

We have presented an argument for why the combinatorial factor graph approach to joint inference can be considered a very general and principled framework for representing and reasoning over

	Template	Instantiated
General	$\{j-i, i, j\}$	LEN-3, START-13, END-16
CCM $CCM(i,j)$	$\{POS(i-1)+POS(j+1)$, $POS(i)+...+POS(j)\}$	OUTER-RB-IN, INNER-DT-VBG-NN
Unigram $U(i)$	$\{Word(i), POS(i)$, $Word(i)+POS(i), Capitalization?(i)$	WORD-shining, TAG-VBG, WORD-POS-shining-VBG, CAP-FALSE
Bigram $B(i,j,w)$	$U(i)\times U(j), U(i)\times U(j+w), U(i-w)\times U(j)$ $POS(i-w)+...+POS(i+w)\times U(j), ...\}$	SPOS0-EPOS0-DT-NN, SPOS+1-EWORD0-VBG-symbol, BG-EW-VBD-RB-DT-VBG-NN-symbol, ...
Variation $V(i,j,w)$	$\{$ $ContainsPOS(l), Contains(w)$, $Tagset(i,j), ConjContains(i,j,w)\}$	CONTAINS-POS-VBG, CONTAINS-WORD-shining TAGSET- DT-VBG-NN, CONTAINS-WORD-SHINING, ...

DT	JJ	NN	IN	WP	VBD	RB	|[DT	VBG	NN]|	IN	DT	NN
the	final	chapter	of	what	was	once	|[a	shining	symbol]|	of	the	future.

Table 1: Common feature templates. Instantiations of these templates on the 13-16 span of the example sentence are also provided. CCM refers to the features used by the constituent-context model (Klein and Manning, 2002).

joint models. In this section we aim to demonstrate that these benefits are not merely conceptual or aesthetic in nature, but translate to practical performance improvements in the model. In the following sections we first demonstrate that parsers based on this architecture outperform previous CRF parsers, and that they provide both an asymptotic advantage over state-of-the-art parsers in decoding speed and an attractive compromise between speed and performance. We then show that the semi-CRF NER model is a comparable baseline to previous standalone models, and that it improves upon previous results when trained jointly.

5.1 Data

We primarily evaluate all of our model configurations on the same data set: a selection of six corpora drawn from the English broadcast news section of the OntoNotes 2.0 dataset (Hovy et al., 2006). The data are partitioned to achieve an approximately 3:1 ratio between training and test sets. This is an exact reproduction of the partitioning found in (Finkel and Manning, 2009), where detailed corpus statistics may be found. As in that work, we remove empty leaf nodes, coarsen nonterminal labels (NP, not NP-PRD), and filter out sentences longer than 40 words. In supplementary parsing experiments we make us of the OntoNotes Wall Street Journal Corpus distribution, using the standard train/test split and sentences with between 3 and 40 words.

5.2 Features

For each of the models presented here, every boolean variable mentioned has a corresponding unary factor representing its likelihood of being true. This leads to a wide-variety of features, depending on the variable's semantics.

For parsing we rely on features comparable to those used in edge-factored dependency parsing (McDonald et al., 2005), consisting of combinations of unigram features (word, part-of-speech, capitalization, and presence of punctuation) between the tokens at or near the span boundaries, including tokens immediately outside and inside of the span. This allows us to capture very strong lexical indications of a span, such as the presence of a comma immediately outside the start and end of the indices. We also look for the occurrence of particular tags anywhere within the span, which might signify that a constituent should not span those indices unless it is sufficiently large. A previous generative model of span-based grammar induction (Klein and Manning, 2002) considered the probability of a span to depend on the part-of-speech tags of the two words immediately outside of it, and on the conjunction of the tags within it. For spans that are sufficiently small (here $w < 10$), the part-of-speech tags of all words inside of the span are concatenated. Examples of these feature sets are provided in Table 1. The reliance on part-of-speech tags as features is afforded through the use of an existing maxent part-of-speech tagger (Toutanvoa and Manning, 2000) which allows us to treat part-of-speech tags as observed while maintaining a fair comparison to other systems which also do not utilize gold part-of-speech tags.

Features for the unary classifier portion of the model also consist of the same unigram features (word, tag, capitalization, etc.), but are taken over 5-word windows from the tokens of the sentence.[5]

Features for NER spans are generally identical to those used for syntactic spans, while the feature sets specifying span labels are much more lexically-oriented. In addition to a small set of contextually-directed information we look primarily at word shape; character n-grams (windows of 2 to 5); capitalization; normalization; regular expressions for initials, numerics, times, and Roman numerals; and membership in a set of lexicons for ordinals, days of the week, months, scientific units, common names, honorifics, and stop words.

5.3 Parsing Results

We present the results of the LBM on labeled parsing, largely in comparison to similar work using CRF parsing with chart decoding (F&M'09). The LBM significantly outperforms the previous standalone approach in all data sets, yielding improvements of up to over nine points in F1 over this model (Table 2). Though the models are generally quite similar, the large performance discrepancy does raise the question of whether or not this could be due solely due to the unique hybrid approach to parsing represented by the LBM over traditional methods, where local factors observe much more information than can be traditionally utilized while still having strong top-down structural constraints at play. It is possible that some of this gap is merely due to different feature sets, but the feature set used here is by no means an exhaustive or fine-tuned set, comparable in size to the features often used by graph-based dependency parsers.

In comparison to other widely available parsers, which are constructed using a vertical and horizontal Markov window of 1 where applicable, the LBM does not achieve absolute state-of-the-art, but still compares favorably to these established models on this data set, outperforming both configurations of the Stanford Parser. It is important to note that these factor graph representations of syntax are still extensible enough to incorporate non-terminal or grammar refinement, or reranking, to further improve their performance. Our purpose in this section is simply to present them as a suitable alternative to existing parsers and prove that the syntactic predictions upon which other tasks can be jointly modeled are themselves quite accurate.

5.4 Decoding Efficiency

We compare against reference implementations of the PCFG parser (Klein and Manning, 2003) and lexicalized parser included in the Stanford Parser distribution[6], and train up comparable grammars for all models from sections 2–21 of the Wall Street Journal Treebank corpus (Marcus et al., 1993).

Efficiency of the LBM is quite competitive with these models (Figure 3). When evaluated in its standard configuration, with a full nonterminal set for each span, the LBM edges out the PCFG

[5]It is also possible to derive features from the parse tree, using the constituent labels collected by traversing the tree along the root-to-leaf path. This results in an average of 4.63% F1 improvement on the unary prediction task over linear features, but requires that the tree be decoded prior to generating unary productions. For convenience we predict the tree and its unary productions simultaneously.

[6]V. 1.68, `http://nlp.stanford.edu/software/lex-parser.shtml`

		Bracket Evaluation					Labeled Evaluation		
Data	Model	Prec	Recall	F1	CB	NC	Prec	Recall	F1
ABC	LBM	82.80	79.65	81.20	1.72	50.78	79.75	76.72	78.20
	+Rules	–	–	–	–	–	80.66	77.92	**79.26**
	F&M '09	–	–	–	2.28	46.88	70.18	70.12	70.15
CNN	LBM	86.40	83.12	84.73	0.96	67.41	83.30	80.14	81.69
	+Rules	–	–	–	–	–	84.66	80.88	**82.72**
	F&M '09	–	–	–	1.11	70.06	76.92	77.14	77.03
MNB	LBM	80.77	76.18	78.41	1.40	59.26	76.98	72.61	74.73
	+Rules	–	–	–	–	–	79.09	73.05	**75.96**
	F&M '09	–	–	–	1.88	59.03	63.97	67.07	65.49
NBC	LBM	80.77	77.37	79.04	1.41	49.66	74.81	71.67	73.20
	+Rules	–	–	–	–	–	76.94	72.34	**74.57**
	F&M '09	–	–	–	2.67	48.92	59.72	63.67	61.63
PRI	LBM	85.01	82.15	83.56	1.44	57.40	82.70	79.92	**81.29**
	+Rules	–	–	–	–	–	81.90	80.44	81.17
	F&M '09	–	–	–	1.72	56.70	76.22	76.49	76.35
VOA	LBM	85.71	81.96	83.80	1.63	43.34	83.55	79.89	81.68
	+Rules	–	–	–	–	–	83.07	82.69	**82.88**
	F&M '09	–	–	–	2.44	38.89	76.56	75.74	76.15
WSJ-Mod	LBM						84.88	80.33	82.54
	+Rules								84.32
	Stanford-PCFG						80.71	79.86	80.28
	Stanford-Factored						81.64	81.65	81.64
	Berkeley						86.61	85.81	**86.21**

Table 2: Labeled model performance. The feature-rich Labeled Bracket Model (LBM) provides significant gains over previously published CRF parsing scores. Unlabeled parsing was evaluated against the true, non-binarized bracketings. The +Rules model refers to the grammar-enhanced version of the model described in Section §6. As in previous work, we find that in the joint setting we find only marginal improvements in parsing F1, and abstain from providing them for the sake of clarity.

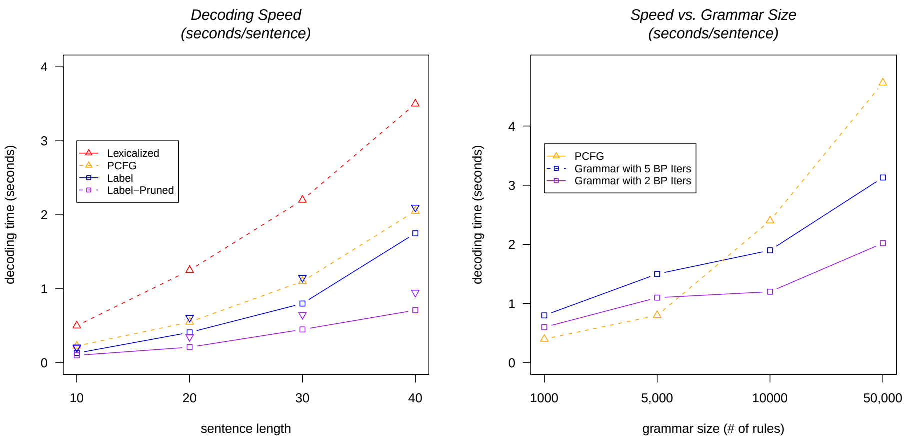

Figure 3: Decoding speeds. Comparing the performance of the labeled model, in both its standard and pruned configuration, with standard PCFG and lexicalized parsing baselines (left), the factor graph models generally decode faster than their counterparts. While featurization cost (inverted triangle annotation) hinder the standard configuration, the pruned model is quite fast. When comparing the grammar model to the PCFG model, and increasing the size of the grammar (right), the additive grammar term provides a clear and substantial benefit as the grammar size increases.

model, the faster of the two baseline systems, but only marginally so. Additionally we indicate the contribution that feature extraction time makes to overall decoding time by the carets above the factor graph parser points. In the case of the standard LBM this removes any performance benefit provided by the factor graph implementation over the PCFG. However, when combined with pruning and reducing the binarized nonterminal set to use one, the LBM becomes a very attractive choice, decoding almost twice as fast on 40-word sentences as the PCFG, and three times faster than the lexicalized model.

The bracket model cannot offer competitive accuracy when compared to the other models and has limited application given its output is only a set of projective spans. However, coupled with an appropriate task it does offer exceptional parsing speed, decoding length 40 sentences at more than 10 per second on our test system, and may be useful as a component in select joint modeling tasks.

5.5 NER Results

We again evaluate on the OntoNotes corpus, which contains both syntactic and named entity annotation, training our model with 10 iterations of stochastic gradient descent (SGD), each with 5 iterations of BP, evaluating on the more difficult full 16-entity label set. Results are presented in Table 3.

Both baseline systems, trained without any syntactic information, perform comparably on average. This is somewhat surprising given the sequential constraints of our NER model, but likely also due in part to the less difficult 4-label NER task reported in previous results. However, we see stronger than average gains by coupling the models and using joint inference. For instance, the factor graph model outperforms the previous best results on four of the six data sets, but improves over it by a much larger margin than the F&M system does on the two data sets it performs best in.

		Prec	Recall	F1	F&M '09
ABC	NER Only	76.4	67.8	72.13	74.51
	NER Joint	76.5	71.3	73.93	**74.91**
CNN	NER Only	75.2	75.0	75.07	75.78
	NER Joint	79.2	79.9	**79.56**	78.70
MNB	NER Only	68.9	70.1	69.50	62.21
	NER Joint	72.7	71.3	**72.02**	66.49
NBC	NER Only	69.5	61.8	65.69	63.90
	NER Joint	73.3	67.0	**70.18**	67.96
PRI	NER Only	80.3	82.6	81.50	83.44
	NER Joint	86.9	86.6	**86.71**	86.34
VOA	NER Only	81.4	74.8	78.11	79.23
	NER Joint	86.4	88.1	87.23	**88.18**

Table 3: NER baseline and joint model performance. Decoding jointly consistently improves the NER results across all corpora, providing the largest gains on corpora with the most data and best parser performance. In comparison to F&M '09, our joint model outperforms in all but two of the corpora.

6 Grammar Rules as Factors

Our final extension, the incorporation of grammar rules into the model, requires only the addition of rule factors that connect triples of labeled span variables:

- Let $\{\text{Rule}(i,j,k,X,Y,Z) : 0 \leq i < j < k \leq n; X,Y,Z \in L\}$, be a sparsely-applied set of ternary factors which coordinate across the $Label$ variables at spans (i,j), (j,k), and (i,k).

The difficulty in constructing a rule-based augmentation is not as much structural as it is about finding an efficient way to add rules to the model. It is easy to enumerate the $O(|L|n^2)$ potentials necessary for the LBM, but it becomes prohibitive to work with the $O(|L|^3n^3)$ potentials that represent a complete grammar.

Figure 4 illustrates a rule factor constraining three $Label$ variables. Grammar rules don't function in the traditional way, where rules guide a set of allowable derivations. Instead, a Rule factor may be instantiated across any triplet of nonterminals to alter the local $Span$ and $Label$ beliefs. Each rule has a single primary feature: its string representation (NP → NP VP), though we do also incorporate back-off features for smoother statistics (X → NP VP, NP → X VP). The ease with which the LBM can be augmented with rules is due to the factorization of the $Label$ variables into large sets of binary variables, instead of one high-dimensional multinomial, for each span.

To learn a sparse set of grammar rules, we used perceptron updates. After decoding a set of training sentences, we computed the difference between the rule applications in the hypothesized and true trees, and updated the weights of those productions accordingly. This creates grammars of roughly 1200 productions on both OntoNotes and WSJ and significantly outperforms the LBM results (Table 2).

Figure 3 illustrates the performance of the grammar model in comparison to a PCFG as we artificially increase the size of the grammar (sentences were fixed at length 40). Our model is unlexicalized and coarse-grained, so simply constructing a grammar by reading off the productions found in the WSJ treebank yields only 2751 rules. While the intent of developing this model is specifically for fast joint inference, lexicalizing the grammar or refining it via a split-merge procedure to bring

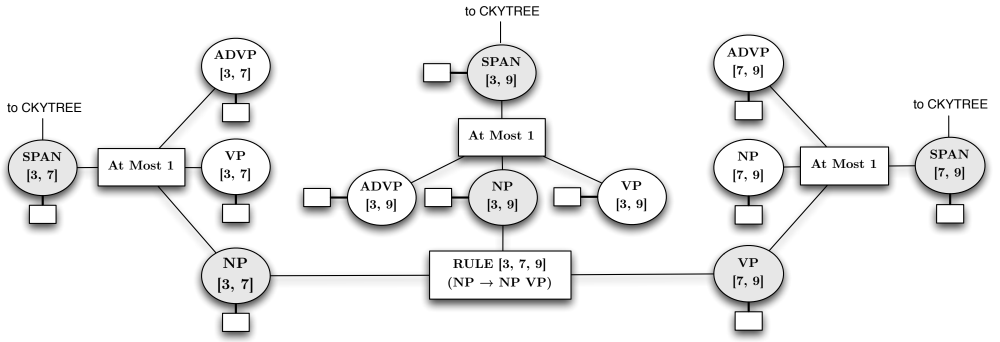

Figure 4: A Rule factor constraining three *Label* variables. Not every span or constituent label must participate in a rule; the number of rules added on top of the labeled bracket model may be quite small. Rules act only to fix up small ungrammatical or dispreferred configurations that are more probable when looking solely at local span information.

performance to state-of-the-art levels solely on parsing increases the grammar size, often producing grammars comprising millions of rules in the latter case. In these cases our model provides clear asymptotic benefits. As depicted in Figure 3 (right), even by 50,000 rules the grammar model has begun to decode significantly faster than the PCFG, with a strong asymptotic advantage with grammars in excess of 10k rules.

All our grammar model performance experiments use five iterations of BP (less if the model converges faster), but for comparison we also present decoding times if we ran the parser as if it were exact inference, and required only two.

6.1 Comparisons to Established Parsers

How do these models handle parsing on larger, more traditionally-sized corpora? On our modified WSJ data set (Table 2, bottom) the LBM performs surprisingly well in comparison to the widely used Stanford and Berkeley parsers (Finkel et al., 2008; Petrov and Klein, 2007), achieving an F1 of 82.54% despite having no representation of a grammar. Augmenting this model with 15 iterations of rule induction yields our largest improvement over the label model, increasing F1 by 1.78% to 84.32% and brings performance in line with state-of-the-art parsers.

Another consideration is our reliance on local potentials, and predictions made primarily based on the rich sources of information at, around, and within span boundaries. While joint tasks may at some point leverage more sophisticated tree annotations, many systems, such as named-entity recognizers, rely primarily on accurately identifying NP spans. Despite having lower labeled F1 than one of the three widely-used constituency parsers we evaluated, the LBM outperforms them all on NP detection (Table 4), making it unlikely that further NER improvements will be had by leveraging these much more sophisticated, but slower, models.

7 Conclusions and Future Work

We have shown how to decompose and represent constituency parsing in terms of variables in a dynamic Markov random field, decoding with general inference algorithms: belief propagation in the exact case for both labeled and unlabeled parsing without a grammar, and loopy BP when augmented with rules or coupled with an NER model. We have demonstrated the convenience

Model	NP-P	NP-R	NP-F1
LBM	90.07	90.02	90.42
+Rules	90.15	91.07	**90.60**
Stanford-PCFG	83.88	85.74	84.80
Stanford-Factored	85.07	87.83	86.43
Berkeley	89.17	90.96	90.06

Table 4: NP prediction results. While some established parsers outperform LBM in general parsing F1 (82.54 LBM vs. 86.21 Berkeley), LBM outperforms all evaluated parsers on a measure more reflective of its potential to many joint modeling scenarios, while remaining asymptotically faster.

and effectiveness of using these combinatorial syntactic representations in joint inference tasks, generally improving performance on named entity recognition over the previous state-of-the-art.

As a standalone parser, both the LBM and rule model provide some decoding efficiency benefit over PCFG and lexicalized baselines, while providing parsing results comparable with the best coarse-grammar, unlexicalized parsers. We also showed how labeling spans via a set of boolean label variables allows ternary factors to function as grammatical rules, not specifying an entire derivation but instead fixing up trees in situations where the LBM would otherwise err.

One natural extension to the parser portion of this work is to port over useful advances from the parsing literature, lexicalizing the grammar or refining the nonterminal set. However, it is also possible to exploit other aspects of the factor graph representation. Most pertinent to parsing performance, a factor graph allows for arbitrary constraints between variables—a potentially promising avenue for incorporating additional linguistic information (headedness, lexicalization, agreement) in unique and powerful ways. This approach would be particularly interesting for languages whose dependencies are hard to capture with the traditional independence assumptions made by PCFGs. The more inherently distributed structure of this model would also make it a good choice for parallelization.

8 Acknowledgements

We thank Jenny Finkel for help in obtaining the additional performance numbers for her system, and Benjamin Börschinger, Mark Johnson, and the anonymous reviewers for their helpful comments. This work was supported in part by the Center for Intelligent Information Retrieval and in part by Army prime contract number W911NF-07-1-0216 and University of Pennsylvania subaward number 103-548106. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Auli, M. and Lopez, A. (2011). A comparison of loopy belief propagation and dual decomposition for integrated CCG supertagging and parsing. In *ACL*, pages 470–480.

Baker, J. K. (1979). Trainable grammars for speech recognition. In *Proceedings of the Acoustical Society of America*, pages 547–550.

Bodenstab, N., Hollingshead, K., and Roark, B. (2011). Unary constraints for efficient context-free parsing. In *ACL*, pages 676–681.

Collins, M. (2003). Head-driven statistical models for natural language parsing. *Comput. Linguist.*, 29(4):589–637.

Finkel, J. R., Kleeman, A., and Manning, C. D. (2008). Efficient, feature-based, conditional random field parsing. In *ACL*, pages 959–967.

Finkel, J. R. and Manning, C. D. (2009). Joint parsing and named entity recognition. In *HLT-NAACL*, pages 326–334.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2006). OntoNotes: The 90% solution. In *HLT-NAACL*, pages 57–60.

Jackendoff, R. (2002). *Foundations of Language: Brain, Meaning, Grammar, Evolution*. Oxford University Press.

Klein, D. and Manning, C. D. (2002). A generative constituent-context model for improved grammar induction. In *ACL*, pages 128–135.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *ACL*, pages 423–430.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP*, pages 523–530.

Naradowsky, J., Riedel, S., and Smith, D. A. (2012). Improving nlp through marginalization of hidden syntactic structure. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing*, Jeju, Korea. Association for Computational Linguistics.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (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. Association for Computational Linguistics.

Petrov, S. and Klein, D. (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, Rochester, New York. Association for Computational Linguistics.

Sarawagi, S. and Cohen, W. W. (2004). Semi-Markov conditional random fields for information extraction. In *NIPS*, pages 1185–1192.

Selkirk, E. (1984). *Phonology and Syntax: The Relation Between Sound and Structure*. Cambridge: MIT Press.

Smith, D. A. and Eisner, J. (2008). Dependency parsing by belief propagation. In *EMNLP*, pages 145–156.

Toutanvoa, K. and Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP*, pages 63–70.

Vaswani, A., Mi, H., Huang, L., and Chiang, D. (2011). Rule markov models for fast tree-to-string translation. In *ACL*, pages 856–864.