Joint A* CCG Parsing and Semantic Role Labeling

Mike Lewis **Luheng He** **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{mlewis,luheng,lsz}@cs.washington.edu

Abstract

Joint models of syntactic and semantic parsing have the potential to improve performance on both tasks—but to date, the best results have been achieved with pipelines. We introduce a joint model using CCG, which is motivated by the close link between CCG syntax and semantics. Semantic roles are recovered by labelling the deep dependency structures produced by the grammar. Furthermore, because CCG is lexicalized, we show it is possible to factor the parsing model over words and introduce a new A* parsing algorithm—which we demonstrate is faster and more accurate than adaptive supertagging. Our joint model is the first to substantially improve both syntactic and semantic accuracy over a comparable pipeline, and also achieves state-of-the-art results for a non-ensemble semantic role labelling model.

1 Introduction

Joint models of syntactic and semantic parsing are attractive; they can potentially avoid the error propagation that is inherent in pipelines by using semantic models to inform syntactic attachments. However, in practice, the performance of joint systems for semantic role labelling (SRL) has been substantially beneath that of pipelines (Sutton and McCallum, 2005; Lluís et al., 2009; Johansson, 2009; Titov et al., 2009; Naradowsky et al., 2012; Lluís et al., 2013; Henderson et al., 2013). In this paper, we present the first approach to break this trend, by building on the close relationship of syntax and semantics in CCG grammars to enable both (1) a simple but highly effective joint model and (2) an efficient A* parsing algorithm.

Semantic dependencies can span an unbounded number of syntactic dependencies, causing significant inference and sparsity challenges for joint models. For example, in the Figure 1, the semantic dependency between *He* and *deny* spans three syntactic edges. This fact makes it difficult to jointly parse syntactic and semantic dependencies with dynamic programs, and means that dependency path features can be sparse. Syntactic dependencies also often have ambiguous semantic interpretations—for example in *He opened the door* and *The door opened*, the syntactic subject corresponds to different semantic roles.

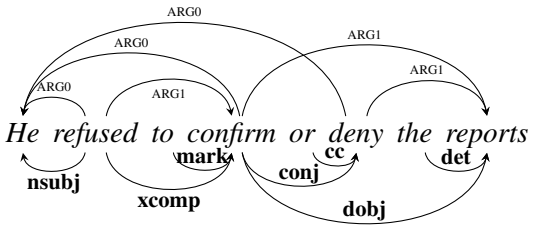


Figure 1: Mismatch between syntactic and semantic dependencies.

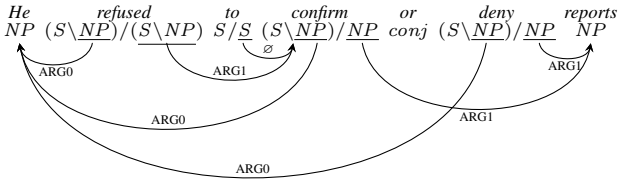


Figure 2: Dependencies produced by a CCG parse. SRL dependencies can be recovered by labelling the edges with a semantic role or ∅. Figure 3 shows a CCG derivation for these dependencies.

We address these challenges with a new joint model of CCG syntactic parsing and semantic role labelling. The CCG formalism is particularly well suited; it models both short- and long-range syntactic dependencies which correspond directly to the semantic roles we aim to recover. The joint model simply involves labelling a subset of these dependencies with the appropriate roles, as seen in Figure 2. This labelling decision can be easily integrated into existing parsing algorithms. CCG also helps resolve cases where interpretation depends on the valency of the pred-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1444–1454,
Lisbon, Portugal, 17-21 September 2015.

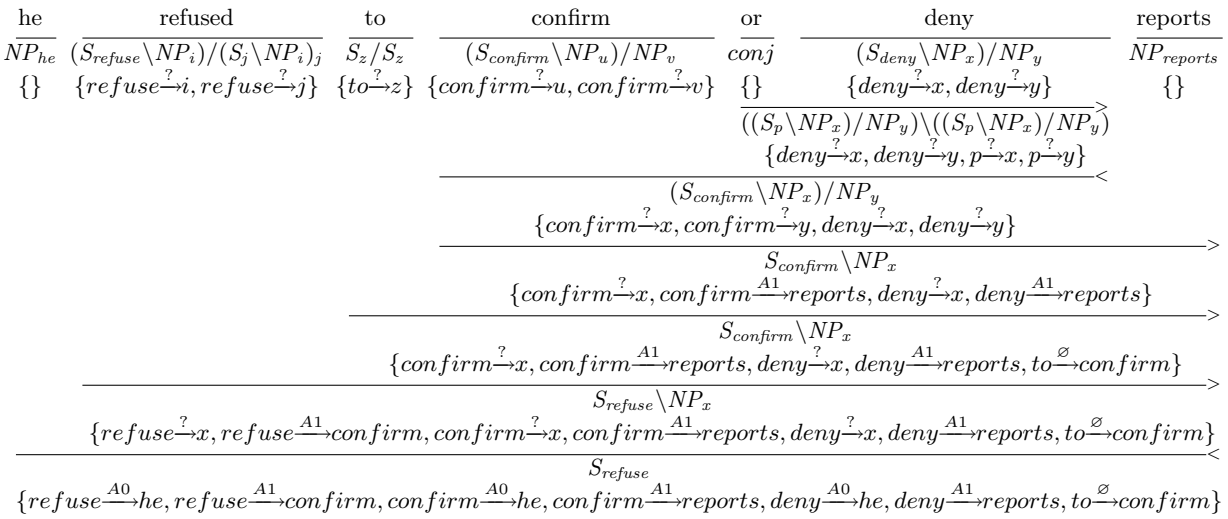


Figure 3: Jointly building a CCG parse and semantic dependencies representation. Subscripts beneath categories denote heads, which are unified when spans combine. One dependency is created for each argument of each lexical category. In our approach, dependency labels are initially underspecified (represented $f \xrightarrow{?} a$) until an attachment is determined by the derivation and a label is chosen by the model.

icate, such as ergative verbs, by learning lexical entries that pair syntactic arguments with semantic roles, such as $open : S\backslash NP_{ARG1}$ and $open : (S\backslash NP_{ARG0})/NP_{ARG1}$. Figure 3 shows a detailed trace of how the example from Figure 2 is parsed with our model.

We also present a new A* algorithm for the joint model. Because CCG is strongly lexicalized, we are able to introduce a new type of *extended lexical entries* that allows us to factor the model over words and develop effective new upper bounds on the Viterbi outside parse score. A* parsing algorithms have previously been developed for models with tree-structured syntactic dependencies (Klein and Manning, 2003; Auli and Lopez, 2011b), and models with no bi-lexical dependencies, including supertag-factored CCGs (Lewis and Steedman, 2014a). We generalize these techniques to SRL-style graph-structured dependencies.

Experiments demonstrate that our model not only outperforms pipeline semantic role labelling models, but improves the quality of the syntactic parser. PropBank SRL performance is 1.6 points higher than comparable existing work, and semantic features improve syntactic accuracy by 1.6 points. Our A* algorithm is 5 times faster than CKY parsing, with no loss in accuracy. The combination of CCG-based joint modelling and A* decoding gives an efficient, accurate, and linguistically principled parser.[1]

[1]The parser is available from: `https://github.com/mikelewis0/EasySRL`

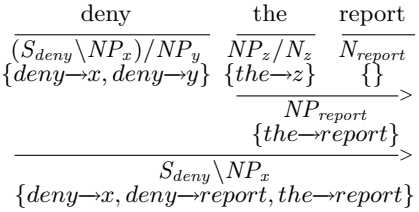
2 Background

2.1 CCG Dependencies

CCG parses define an implicit dependency graph, by associating each argument of each category with a dependency. In contrast to Stanford dependency trees, CCG dependency graphs can be non-projective, and words can have multiple parents. For example, in Figure 2, *He* is an argument of *refuse*, *confirm* and *deny*.

To create dependencies, categories are marked with headedness information—which we denote with subscripts. For example, in the category $(S_{deny}\backslash NP_x)/NP_y$, the final head for the sentence S will be *deny*, and that two dependencies will be introduced from *deny* to unspecified arguments x and y. During parsing, variables are unified with heads, creating fully specified dependencies.

Re-using variables allows dependencies to propagate. For example, the determiner category NP_i/N_i marks that the head of the resulting NP is equal to the head of its N argument (e.g. the head of *the report* is *report*), as in the following parse:



The same mechanism allows long-range arguments to be propagated. Figure 3 shows several long-range arguments, such as how co-indexation

propagates the subject of *deny* to *he*. For more details, see Hockenmaier (2003).

2.2 A* CCG parsing

A* parsing searches for an optimal parse, without building a complete chart (in contrast to CKY parsing). This is particularly attractive for CCG, because the formalism's lexical and derivational ambiguity causes the parse charts to be very dense, in practice. Lewis and Steedman (2014a) showed that A* CCG parsing can be highly efficient, but used a restricted model without bi-lexical features.

In A* parsing, entries y are added to the chart in order of their cost $f(y) = g(y) + h(y)$, where $g(y)$ is the inside score for the entry, and $h(y)$ is an upper bound on the Viterbi outside score. Because partial parses are built in order of increasing $f(y)$, the first complete parse added to the chart is guaranteed to be optimal. The key to making A* parsing efficient is computing tight upper bounds on the outside score. In Lewis and Steedman's supertag-factored model, the bound can be computed as the sum of the highest-scoring supertags in the outside parse.

The agenda is initialized with items representing every category for every word. Then, after each item is added to the chart, the agenda is updated with all binary and unary rules that can be applied to the new item. For more details, see Lewis and Steedman (2014a).

3 Model

Our joint model of CCG and SRL parsing simply involves labelling CCG syntactic dependencies (which are implicit in CCG parses), with SRL roles (or null). This formulation allows us to easily adapt the log-linear CCG parsing model of Clark and Curran (2007) to the joint setting by working with extended dependencies that including syntactic and semantic information.

More formally, we can define a notion of consistency to specify the labelling of syntactic dependencies. A set of semantic dependencies π is consistent with a CCG derivation d if each semantic dependency corresponds to a single syntactic dependency—for example, see the dependencies in Figure 3. The CCG derivation in Figure 3 would also be consistent with the SRL dependency $refuse \xrightarrow{ARG2} he$, but not $refuse \xrightarrow{ARG1} reports$ (because the derivation does not produce the required syntactic dependency).

Now, we can define a log-linear model over pairs of consistent CCG derivations d and SRL dependencies π:

$$p(d, \pi|x) = \frac{e^{\theta \cdot \phi(x,d,\pi)}}{\sum_{(d',\pi') \in GEN(x)} e^{\theta \cdot \phi(x,d',\pi')}}$$

where $GEN(x)$ is the space of consistent pairs for sentence x.

3.1 Dependencies

Because we enforce consistency, we can work with joint dependencies that are a combination of CCG and SRL dependencies. We denote a dependency as a 6-tuple of functor f, lexical category c, argument number n, preposition p, argument a and semantic role r: $\langle f, c, n, p, a, r \rangle$. The first three of these (f,c,n) are lexically specified, but a and r are lexically underspecified until the attachment is determined by the derivation, and a label is chosen by the model. For example, in Figure 3, the lexical entry for *deny* has two underspecified dependencies: $\langle deny, (S\backslash\mathbf{NP})/NP, 1, \varnothing, ?, ? \rangle$ and $\langle deny, (S\backslash NP)/\mathbf{NP}, 2, \varnothing, ?, ? \rangle$.[2] At the end of the derivation, the dependencies are specified as: $\langle deny, (S\backslash\mathbf{NP})/NP, 1, \varnothing, he, ARG0 \rangle$ and $\langle deny, (S\backslash NP)/\mathbf{NP}, 2, \varnothing, reports, ARG1 \rangle$.

The preposition p is lexically underspecified for PP arguments, but otherwise $\varnothing$. Prepositions are marked lexically on $PP/*$ categories, and then propagated, for example:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\text{fly}\\ (S_{fly}\backslash NP_x)/PP_y^p\\ \{fly \xrightarrow{?} x, fly \xrightarrow{?} y\}\end{array} \quad \dfrac{\begin{array}{c}\text{to}\\ PP_z^{to}/NP_z\\ \{to \xrightarrow{?} z\}\end{array} \quad \begin{array}{c}\text{Lisbon}\\ NP_{Lisbon}\\ \{\}\end{array}}{PP^{to}_{Lisbon}\ \{to \xrightarrow{\varnothing} Lisbon\}}<}{S_{fly}\backslash NP_x\ \{fly \xrightarrow{?} x, fly \xrightarrow{ARG1} Lisbon, to \xrightarrow{\varnothing} Lisbon\}}<$$

In the above example, the dependency from *fly* to *Lisbon* corresponds to the tuple: $\langle fly, (S\backslash NP)/\mathbf{PP}, 2, to, Lisbon, ARG1 \rangle$. Propagating both the noun and preposition from prepositional phrases allows the model to use both for features, and improved results.

3.2 Features

We use the following feature classes, which decompose over categories, dependencies, and local rule instantiations.

[2] Bold-face is used to highlight the argument of a category corresponding to a dependency. For example $(S\backslash\mathbf{NP})/NP$ denotes the first (subject) dependency of a transitive verb.

3.2.1 Supertagging Features

Supertagging features ϕ^{CAT} score categories for words. A single feature is used, which is the (unnormalized) score from Lewis and Steedman (2014b)'s supertagging model. The supertagger outputs a distribution over categories for each word independently. The model is trained on supertags extracted from an adaptation of CCGrebank (Honnibal et al., 2010). The adaptation makes minor changes to better match PropBank.

3.2.2 Preposition Features

Preposition features ϕ^{PP} score whether the nth argument of the word at index f with category c should take a PP argument headed by preposition p. We use features $x_f{+}p$, $l_f{+}c$ and $l_f{+}c{+}n{+}p$, where x_f is the fth word, and l_f is its lemma.

3.2.3 Labelling Features

Labelling features ϕ^{ROLE} determine whether the nth argument of a word with lemma l, and category c should take a role r. We use $n{+}r$, $c{+}n{+}r$, $c{+}n{+}p{+}r$, $l{+}c{+}n{+}p{+}r$, $n{+}r$, r, $l{+}p{+}r$, $c{+}n{+}r$, $l{+}r$, $h{+}r$, where h is an indicator of whether l is hyphenated, and p is the preposition of PP arguments.

3.2.4 Dependency Features

Dependency features ϕ^{DEP} score dependencies, based on the functor index f, argument index a and role r. We use $l_f{+}r{+}l_a$, $l_f{+}r{+}c_a$, $d{+}r$, $c_{f+o}{+}r$, $c_{a+o}{+}r$, where o is an offset from $-3 \ldots +3$, d is the distance between f and a, c_i is a cluster or POS tag for word i, and l_i is the lemma of word i. Clusters were created from pre-trained word embeddings (Levy and Goldberg (2014)) using k-means, with $k = 20, 50, 200, 1000, 2500$. These features only apply when the role $r \neq \varnothing$.

3.2.5 Derivation Features

Derivation features ϕ^{DERIV} score properties of the syntactic derivation. To simplify computation of the upper bounds for the A* parsing algorithm given in Section 5, the weights of these features are constrained to be ≤ 0. For simplicity, we only use features for unary rules—for example, a feature records when the unary rule $N \rightarrow NP$ converts a determiner-less N to a NP.

4 Lexical Factorization

In this section, we show how to factor the model into *extended lexical entries*. This factorization allows us to efficiently compute upper bounds on the scores of partial parses, which is crucial to the A* algorithm described in Section 5.

The key observation is that our supertagging, labelling and dependency features can each be associated with exactly one word (using the functor for the dependency features). Therefore, we can equivalently view the complete parse as a series of *extended lexical entries*: $y = y_0 ... y_N$. Extended lexical entries are composed of a lexical category, and a role and attachment for each argument of the category. The set of extended lexical entries specifies the yield of the parse. For example, the parse from Figure 2 can be represented as the following extended lexical entries:

he	$\vdash NP$
refused	$\vdash (S\backslash NP_{ARG0=he})/(S\backslash NP)_{ARG1=confirm}$
to	$\vdash S/S_{\varnothing=confirm}$
confirm	$\vdash (S\backslash NP_{ARG0=he})/NP_{ARG1=reports}$
or	$\vdash conj$
deny	$\vdash (S\backslash NP_{ARG0=he})/NP_{ARG1=reports}$
reports	$\vdash NP$

The score for a sentence x and joint SRL-CCG parse y, $\theta \cdot \phi(x, y)$, can be decomposed into scores of its extended lexical entries y_i, plus a score for the derivation features:

$$\theta \cdot \phi(x, y) = \sum_{y_i \in y} \theta \cdot \phi(x, y_i) + \theta \cdot \phi^{DERIV}(x, y)$$

The space of possible extended lexical entries for word x_i is defined by $GENLEX(x_i)$, which is expressed with a CFG, as shown in Figure 4a.

The features defined in Section 3.2 decompose over the rules of $GENLEX$—so it can be weighted with the globally trained feature weights.

Expressing $GENLEX(x_i)$ as a CFG allows us to efficiently compute upper bounds on the score of y_i when it is only partially specified, using the Viterbi algorithm—which we will make use of in Section 5. Making the attachment choice independent of the syntactic category greatly improves the efficiency of these calculations.

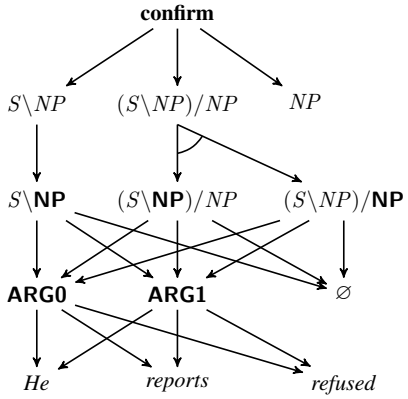
5 A* Parsing

To efficiently decode our model, we introduce a new A* parsing algorithm. As discussed in Section 2.2, the key to efficient A* parsing is computing tight upper bounds on the Viterbi outside score of partial parses, with the function h.

The intuition behind our algorithm is that we can use the lexical factorization of Section 4 to compute upper bounds for words individually,

Lexical Category Choice	
x_i	$\rightarrow NP \mid S\backslash NP \mid (S\backslash NP)/PP \mid \ldots$
One dependency is created for every argument of the category	
$S\backslash NP$	$\rightarrow S\backslash \mathbf{NP}$
$(S\backslash NP)/PP$	$\rightarrow (S\backslash \mathbf{NP})/PP,\ \ (S\backslash NP)/\mathbf{PP}$
$\ldots$	
Preposition choice for PP arguments	
$(S\backslash NP)/\mathbf{PP}$	$\rightarrow (S\backslash NP)/\mathbf{PP^{in}} \mid (S\backslash NP)/\mathbf{PP^{for}} \mid \ldots$
$\ldots$	
Semantic role label choice for the argument	
$S\backslash \mathbf{NP}$	$\rightarrow S\backslash \mathbf{NP_{ARG0}} \mid S\backslash \mathbf{NP_{ARG1}} \mid \ldots$
$(S\backslash \mathbf{NP})/PP$	$\rightarrow (S\backslash \mathbf{NP_{ARG0}})/PP \mid (S\backslash \mathbf{NP_{ARG1}})/PP \ldots$
$(S\backslash NP)/\mathbf{PP^X}$	$\rightarrow (S\backslash NP)/\mathbf{PP^X_{ARG0}} \mid (S\backslash NP)/\mathbf{PP^X_{ARG1}} \cdots$
$\ldots$	
Attachment choice	
$ARG0$	$\rightarrow x_0 \mid \ldots \mid x_{i-1} \mid x_{i+1} \mid \ldots \mid x_N$
$ARG1$	$\rightarrow x_0 \mid \ldots \mid x_{i-1} \mid x_{i+1} \mid \ldots \mid x_N$
$\ldots$	

(a) The grammar $GENLEX(x_i)$



(b) Visualization of a fragment of $GENLEX(confirm)$

Figure 4: (a) The grammar $GENLEX(x_i)$, which defines the space of extended lexical entries, and (b) a visualization of a fragment of $GENLEX(confirm)$. Extended lexical entries, including **confirm** $\vdash (S\backslash NP_{ARG0=he})/NP_{ARG1=reports}$ and **confirm** $\vdash NP$, are specified by choosing one category (top level in both a and b), enumerating all arguments (second level), selecting the preposition for PP arguments (when present), selecting a semantic role label for each, and finally choosing the argument head word. The features are local to the grammar rules, enabling efficient dynamic programs for upper bound computations on partially specified entries, such as **confirm** $\vdash (S\backslash NP_{r=a})/NP_{ARG1=reports}$.

then create an upper bound for the parse as a sum of upper bounds for words. The bound is not exact, because the grammar may not allow the combination of the best lexical entry for each word.

Section 5.1 gives a declarative definition of h for any partial parse, and 5.2 explains how to efficiently compute h during parsing.

5.1 Upper Bounds for Partial Parses

This section defines the upper bound on the Viterbi outside score $h(y_{i,j})$ for any partial parse $y_{i,j}$ of span $i \ldots j$. For example, in the parse in Figure 3, $y_{3,5}$ is the partial parse of *confirm or deny* with category $(S\backslash NP)/NP$.

As explained in Section 4, a parse can be decomposed into a series of extended lexical entries. Similarly, a partial parse can be viewed as a series of *partially specified* extended lexical entries $y_0 \ldots y_N$. For example, in Figure 3, the partial parse of the span *confirm or deny reports*, the extended lexical entries for the words outside the span (*He*, *refused* and *to*) are completely unspecified. The extended lexical entries for words inside the span have specified categories, but can contain underspecified dependencies:

confirm	$\vdash (S\backslash NP_{r=a})/NP_{ARG1=reports}$
or	$\vdash conj$
deny	$\vdash (S\backslash NP_{r'=a'})/NP_{ARG1=reports}$
reports	$\vdash NP$

Therefore, we can compute an upper bound for the outside score of a partial parse as a sum of the upper bounds of the unspecified components of each extended lexical entry. Note that because the derivation features are constrained to be ≤ 0, they do not affect the calculation of the upper bound.

We can then find an upper bound for completely unspecified spans using by summing the upper bounds for the words. We can pre-compute an upper bound for the span $h_{i,j}$ for every span i, j as:

$$h_{i,j} = \sum_{k=i}^{j} \max_{y_k \in GENLEX(x_k)} \theta \cdot \phi(x, y_k)$$

The max can be efficiently computed using the Viterbi algorithm on $GENLEX(x_k)$ (as described in Section 4).

The upper bound on the outside score of a partial parse is then the sum of the upper bounds of the words outside the parse, and the sum of the scores of the best possible specifications for each underspecified dependency:

$$h(y_{i,j}) = \sum_{\langle f,c,n,?,?,? \rangle \in \mathbf{deps}(y_{i,j})} \max_{a',p',r'} \theta \cdot \phi(\langle f, c, n, p', a', r' \rangle) + h_{0,i-1} + h_{j+1,N}$$

where $\mathbf{deps}(y)$ returns the underspecified dependencies from partial parse.

For example, in Figure 3, the upper bound for the outside score of the partial parse of *confirm or deny reports* is the sum of the upper bounds

for the other words independently ($h_{0,3}$) added to the score of the best attachments and roles for the subjects of *confirm* and *deny* independently.

5.2 Additive Updates

During parsing, the upper bound h can be efficiently computed recursively with additive updates. Initially h is the sum of the upper bounds for each word independently $h_{0,N}$. Then, when specifying categories or labelling dependencies, the score is updated.

- When specifying a category for a word x_i, the Viterbi score of $GENLEX(x_i)$ is subtracted from h, and the sum of Viterbi scores for each of the category's (underspecified) dependencies is added to h.
- When specifying a semantic role for a dependency with functor f, causing the dependency to be fully specified, the Viterbi score for the node representing that dependency in $GENLEX(x_f)$ is subtracted from h.
- When a binary rule combines two partial parses $y_{i,j}$ and $y_{j+1,k}$, the bound on the outside score is updated by summing the bounds of the outside scores of the child parses, and subtracting the overall upper bound for the sentence (to avoid double-counting):
  $h(y_{i,k}) = h(y_{i,j}) + h(y_{j+1,k}) - h_{0,N}$
  This update can be derived from the definition of h.

These are the only cases where h is updated.

6 Training

During training, we optimize parameters θ for the marginal likelihood of the semantic dependencies π_i given a sentence x_i, treating syntactic derivations d as a latent variable and using L_2 regularization.

$$L(\theta) = \log \prod_i p_\theta(\pi_i|x_i) - \frac{\sum_j \theta_j^2}{2\sigma^2}$$
$$= \sum_i \log \frac{\sum_{d \in \Delta(x_i,\pi_i)} e^{\theta \cdot \phi(x_i,d,\pi_i)}}{\sum_{(d',\pi') \in GEN(x_i)} e^{\theta \cdot \phi(x_i,d',\pi')}} - \frac{\sum_j \theta_j^2}{2\sigma^2}$$

where $GEN(x_i)$ is the set of consistent CCG and SRL parses for a sentence x_i (see Section 3), and $\Delta(x_i, \pi_i)$ is the set of CCG derivations that are maximally consistent with gold SRL parses π_i. More formally, if $\textbf{labelled}(y)$ returns the set of labelled dependencies from parse y, then:

$$\Delta(x, \pi) = \arg\max_{y \in GEN(x)} |\pi \cap \textbf{labelled}(y)|$$

The arguments of PropBank dependencies can span multiple words, so CCG dependencies are marked as equivalent if their argument is anywhere within the PropBank span.

The approach is closely related to the hybrid dependency model (Clark and Curran, 2007). However the CCGbank dependencies used by Clark and Curran's model constrain all lexical and attachment decisions (only allowing 'spurious' derivational ambiguity) whereas our use of semantic dependencies models most of the syntactic parse as latent.

6.1 Hyperparameters

Calculating the gradient of the loss function requires computing the marginal probability of the correct parses. Computing marginal probabilities exactly involves summing over all possible CCG parses, which is intractable. Instead, following Clark and Curran, we limit the size of training charts using a variable-width supertagger beam β, initially 10^{-3}. If the chart size exceeds 100000 nodes, we double β and re-parse. For computing Δ, we use a more restrictive beam of $\beta = 0.01$ to improve the quality of the positive training examples. We optimize using L-BFGS (Liu and Nocedal, 1989), with $\sigma^2 = 0.06$.

6.2 Pruning

To improve efficiency, we compute a number of thresholds, by aligning gold CCGbank dependencies with PropBank. If an argument of a category occurs with a particular semantic role less than 3 times in the aligned data, it is pruned from the training and decoding charts. We also filter infrequent features before training. We count the number of times each feature occurs in the aligned data, and filter features occurring less than 3 times. During decoding, we prune lexical categories whose probability under the supertagging model is less than a factor of 10^{-2} of that of the best category for that word. If the chart size exceeds 20000 nodes, we back off to a pipeline model (roughly 3% of sentences). Finally, we build and use a tag dictionary in the same way as Lewis and Steedman (2014a).

7 Experiments

7.1 Experimental Setup

We used PropBank Section 00 for development, Sections 02-21 for training, and Section 23 for testing. The **Pipeline** baseline first parses with a supertag-factored A* model, and chooses a semantic role for each CCG dependency with a log-linear classifier. The classifier uses the role and attachment features used by the parser.

7.2 Semantic Role Labelling

We evaluate our parser as a dependency-based SRL model on PropBank, comparing with CoNLL-2008 systems (Surdeanu et al., 2008).

Comparison systems Following Täckström et al. (2015), we compare with the best 'single parser' SRL models, which use only a single syntactic parse.[3]

Much recent work has evaluated using gold predicate identification (Hajič et al., 2009; FitzGerald et al., 2015). This setting is particularly unrealistic for our joint model, where gold predicate identification would be a highly useful feature for the supertagger; we only compare with models that use automatic predicate identification. To the best of our knowledge, the best SRL results with automatic predicate identification were achieved at CoNLL-2008.

A number of other models have been evaluated on CoNLL-2008 data. While we cannot compare directly on our metric, the best reported joint model (Johansson, 2009) scored 1.4 points lower than the Che et al. (2008) system we compare to on the CoNLL metric on PropBank. Other joint models, such as those of Titov et al. (2009) and Lluís et al. (2013), achieve similar performance.

Evaluation Metric Comparing fairly with existing work is complicated by the mismatch between heads found by our model and those used in other evaluations. Headedness decisions are often arbitrary—for example, whether a prepositional phrase is headed by the noun or preposition—and different choices were made in the design of CCGbank and the CoNLL-2008 headedness rules.

To solve this problem, we introduce a new *within-constituent* metric, which awards dependencies as correct if they attach anywhere within the original PropBank-annotated argument spans. For example, if the PropBank annotates that the *ARG*0 of *owned* is *by Google*, a dependency to either *by* or *Google* is judged correct. We compute new scores for the CoNLL-2008 submissions on our metric, filtering reference and continuation arguments (which are artifacts of the CoNLL conversion of PropBank, but not required by our metric), and nominal predicates based on POS tag. The ranking of the top 5 CoNLL-2008 open-track models is identical under our metric and the original one (up to statistical significance), suggesting that our metric is equally discriminative. However, perhaps interestingly, the ranking of Vickrey and Koller (2008) does improve—likely due to the use of a syntactic formalism with different headedness rules. For simplicity, we do not include predicate senses in the evaluation.

[3]The best models use reranking with powerful global features (Toutanova et al., 2008; Johansson and Nugues, 2008) or ensemble methods (Surdeanu et al., 2007; Punyakanok et al., 2008). These techniques have the potential to improve any SRL system, including ours, at some expense in speed.

	PropBank			Brown		
Model	P	R	F1	P	R	F1
Vickrey	**87.3**	77.3	82.0	**74.0**	64.5	68.9
Che	85.3	78.6	81.8	71.1	65.7	68.0
Zhao	82.4	79.8	81.1	66.6	64.9	65.7
Riedel	83.6	74.7	78.9	69.3	62.7	65.8
Pipeline	79.2	73.9	76.4	69.3	64.0	66.1
Joint	84.8	**82.2**	**83.5**	71.2	**69.2**	**70.2**

Table 1: Comparison with the best single-parser SRL models on PropBank from CoNLL-2008. The comparison models are Vickrey and Koller (2008), Che et al. (2008), Zhao and Kit (2008) and Riedel and Meza-Ruiz (2008).

Results are given in Table 1 and show that our joint model greatly outperforms the pipeline version, demonstrating the value of joint reasoning. It also scores 1.5 points higher than the best comparable models in-domain, and 1.3 points higher out-of-domain. To the best of our knowledge, this is the first joint syntax/SRL model to outperform strong pipeline models.

7.3 Efficiency Experiments

We explore whether our A* decoding is more efficient than alternatives, including both algorithmic and feature computation. While the A* search builds very small charts, the features must be precomputed for the heuristic. In CKY parsing, features can be computed lazily.

Model	Sentences per second	PropBank F1
Pipeline A*	**38.3**	76.4
Joint CKY	6.0	**83.5**
Joint AST	20.3	83.0
Joint A*	31.3	**83.5**

Table 2: Parser speed on PropBank Section 23

We compare with a CKY parsing over the same space. If no parse is found, or the chart size exceeds 400000 nodes, we back off to the pipeline (tuned so that the backoff is used roughly as often as for the A* parser). We also compare with adaptive supertagging (AST, Clark and Curran (2007)), which is the same except for first attempting to parse with a restrictive supertagger beam $\beta = 0.1$.

Table 2 shows the results. The A* pipeline is fast, but inaccurate. CKY is 5 times slower than A* in the same space, whereas adaptive supertagging trades accuracy for speed. The best previously reported speed improvement using A* parsing is a factor of 1.2 times faster (Auli and Lopez, 2011b). Our new A* algorithm dominates existing alternatives in both speed and accuracy.

7.4 Syntactic Parsing

We also evaluate our model for CCG parsing accuracy, using CCGrebank (Honnibal et al., 2010), and comparing with a C&C parser model adapted to this dataset. Results are shown in Table 3. Of course, given our latent syntax, and the fact we have no model of CCGbank dependencies, we do not expect state-of-the-art accuracy. However, the 1.6 point improvement over the pipeline shows that SRL dependencies are useful for disambiguating syntactic attachment decisions. Many errors were caused by disagreements between CCGbank and PropBank—PropBank is likely to be more accurate as it was hand-annotated, rather than automatically converted from the Penn Treebank. In effect, our latent model of syntax is successfully learning a grammar that better produces the correct semantics.

Table 4 shows the syntactic dependencies which are improved most by modelling semantics. Unsurprisingly, verb arguments and adjuncts relations show large improvements. However, we also see more accurate attachment of relative clauses. While we do not model relative clauses explicitly, correctly attaching them is essential for propagating certain verb arguments (e.g. the subject of *broke* in *the glass on the table that broke*).

Model	P	R	F1
C&C	81.8	81.2	81.5
Pipeline	76.6	77.7	77.2
Joint	78.3	79.4	78.8

Table 3: Labelled F1 for CCGbank dependencies on CCGrebank Section 23

Dependency	Type	Δ F1
$((S_{dcl}\backslash NP)/\mathbf{PP})/NP$		+12.6
$((S_{dcl}\backslash NP)/PP)/\mathbf{NP}$		+11.0
$((S_b\backslash NP)/\mathbf{PP})/NP$	Verb	+10.7
$(S_{dcl}\backslash NP)/\mathbf{PP}$	arguments	+8.1
$(S_{ng}\backslash \mathbf{NP})/NP$		+7.6
$(S_{pt}\backslash \mathbf{NP})/NP$		+6.8
$((S\backslash NP)\backslash(\mathbf{S}\backslash\mathbf{NP}))/NP$	Verb	+16.9
$((S\backslash NP)\backslash(\mathbf{S}\backslash\mathbf{NP}))/S_{dcl}$	adjuncts	+11.9
$(N\backslash\mathbf{N})/(S_{dcl}\backslash NP)$	Relative	+12.5
$(NP\backslash\mathbf{NP})/(S_{dcl}\backslash NP)$	clauses	+8.5

Table 4: CCG syntactic dependencies with the largest change in F1 between the Pipeline and Joint models (of those occurring at least 100 times in the development set).

7.5 Error Analysis

Table 5 gives an analysis of recall errors from the first 100 sentences of PropBank Section 00. One third of errors were syntactic attachment errors. A further 14% were triggered by cases where the parser found the correct attachment, but gave an adjunct a core argument CCG category (or vice versa). Many of these decisions are not obvious—for example in *rose sharply*, PropBank considers *sharply* to be an argument and not an adverb. 21% of errors were caused by choosing the wrong SRL

Error	Percentage
Attachment error	33%
Correct attachment, wrong label	21%
Correct attachment, unlabelled	20%
Argument/adjunct distinction	14%
Problematic constructions	9%
Dubious annotation	4%

Table 5: Error analysis of recall errors from 100 development set sentences.

label. Another 20% were caused by predicates assigning arguments the null semantic role ∅. The major cause of these errors is predicates that act syntactically like adjectives (e.g. *publishing* in *Dutch publishing group*) where syntactic cues are weak. Finally, 9% involved long-range arguments that our current grammar is unable to project. One common case is constructions like *X has a plan to buy Y*, where the grammar does not propagate the subject of *buy* to *X*. Further improvements to CCGbank may help to resolve these cases.

8 Related Work

Joint syntactic and SRL models There have been many proposals for jointly parsing syntactic and semantic dependencies. Lluís et al. (2013) introduce a joint arc-factored model for parsing syntactic and semantic dependencies, using dual-decomposition to maximize agreement between the models. SRL performance is slightly worse than a pipeline version. Naradowsky et al. (2012) introduce a SRL model with latent syntax representations, by modelling a latent dependency tree during training, which is marginalized out at test time. However, performance at English SRL is roughly 7 points beneath state of the art. Other notable models include those of Johansson (2009) and Titov et al. (2009).

CCG parsing Our log-linear model is closely related to that of Clark and Curran (2007), but we model SRL dependencies instead of CCG dependencies. The best CCG parsing results were achieved by Auli and Lopez (2011a), who, like us, score CCG parses based jointly on supertagging and dependency model scores. Decoding their model requires dual-decomposition, to maximize agreement between the separate models. We avoid the need for this technique by using a unigram supertagging model, rather than a sequence model.

CCG semantics Work on semantic parsing has mapped sentences onto semantic representations with latent CCGs (Zettlemoyer and Collins, 2009; Kwiatkowski et al., 2010; Kwiatkowski et al., 2013) for restricted domains. Recent work has scaled these techniques to wide-coverage datasets (Artzi et al., 2015). Krishnamurthy and Mitchell (2014) also explore joint CCG syntactic and semantic parsing. They use a smaller semantic lexicon, containing 130 predicates, rather than the 3257 PropBank verbs. In contrast to our results, jointly modelling the semantics lowers their model's syntactic accuracy.

Other CCG-based SRL models haved used CCG dependencies as features for predicting semantic roles (Gildea and Hockenmaier, 2003; Boxwell et al., 2009), but performance is limited by relying on 1-best parses—a problem we resolved with a joint model.

A* parsing A* parsing has previously been explored for less general models than ours. Klein and Manning (2003) and Auli and Lopez (2011b) use A* parsing for models with tree-structured dependencies. The best reported speed improvement is parsing 1.2 times faster, whereas we improve by a factor of 5. Our model also allows the more complex graph-structured dependencies required for semantic role labelling. Lewis and Steedman (2014a) demonstrate an efficient A* algorithm for CCG, but cannot model dependencies.

9 Conclusions and Future Work

We have shown that using CCG can allow joint models of syntax and semantics to outperform pipelines, and achieve state-of-the-art results on PropBank SRL. Our new A* parsing algorithm is 5 times faster than CKY parsing, without loss of accuracy. Using latent syntax allows us to train the model purely from semantic dependencies, enabling future work to train against other annotations such as FrameNet (Baker et al., 1998), Ontonotes (Hovy et al., 2006) or QA-SRL (He et al., 2015). The semantic labels provided by PropBank can also be integrated into wide-coverage CCG semantic parsers (Bos, 2008; Lewis and Steedman, 2013) to improve performance on downstream applications.

Acknowledgements

This research was supported in part by the NSF (IIS-1252835), DARPA under the DEFT program through the AFRL (FA8750-13-2-0019), an Allen Distinguished Investigator Award, and a gift from Google. We thank Nicholas Fitzgerald, Dan Garrette, Kenton Lee, Swabha Swayamdipta, Mark Yatskar and the anonymous reviewers for their helpful comments on earlier versions of this paper, and Matthew Honnibal for useful discussions.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG Semantic Parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Michael Auli and Adam Lopez. 2011a. A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Michael Auli and Adam Lopez. 2011b. Efficient CCG parsing: A* versus Adaptive Supertagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*.

Johan Bos. 2008. Wide-coverage Semantic Analysis with Boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics.

Stephen A. Boxwell, Dennis Mehay, and Chris Brew. 2009. Brutus: A Semantic Role Labeling System Incorporating CCG, CFG, and Dependency Features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*.

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, and Sheng Li. 2008. A cascaded Syntactic and Semantic Dependency Parsing System. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Stephen Clark and James R Curran. 2007. Wide-coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4).

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic Role Labeling with Neural Network Factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Daniel Gildea and Julia Hockenmaier. 2003. Identifying Semantic Roles Using Combinatory Categorial Grammar. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multi-lingual Joint Parsing of Syntactic and Semantic Dependencies with a Latent Variable Model. *Computational Linguistics*.

Julia Hockenmaier. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh. College of Science and Engineering. School of Informatics.

M. Honnibal, J.R. Curran, and J. Bos. 2010. Rebanking CCGbank for Improved NP Interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% Solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based Syntactic-Semantic Analysis with Propbank and Nombank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Richard Johansson. 2009. Statistical Bistratal Dependency Parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*.

Dan Klein and Christopher D Manning. 2003. A* Parsing: Fast Exact Viterbi Parse Selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*.

Jayant Krishnamurthy and Tom M. Mitchell. 2014. Joint Syntactic and Semantic Parsing with Combinatory Categorial Grammar. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing Probabilistic CCG Grammars from Logical Form with Higher-order Unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-Fly Ontology Matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Mike Lewis and Mark Steedman. 2013. Combined Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics*, 1.

Mike Lewis and Mark Steedman. 2014a. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Mike Lewis and Mark Steedman. 2014b. Improved CCG parsing with Semi-supervised Supertagging. *Transactions of the Association for Computational Linguistics*.

D. C. Liu and J. Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Math. Program.*, 45(3).

Xavier Lluís, Stefan Bott, and Lluís Màrquez. 2009. A Second-order Joint Eisner Model for Syntactic and Semantic Dependency Parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint Arc-factored Parsing of Syntactic and Semantic Dependencies. *Transactions of the Association of Computational Linguistics – Volume 1*, pages 219–230.

Jason Naradowsky, Sebastian Riedel, and David A. Smith. 2012. Improving NLP Through Marginalization of Hidden Syntactic Structure. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics*, 34(2).

Sebastian Riedel and Ivan Meza-Ruiz. 2008. Collective Semantic Role Labelling with Markov Logic. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination Strategies for Semantic Role Labeling. *Journal of Artificial Intelligence Research*, 29(1).

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Charles Sutton and Andrew McCallum. 2005. Joint Parsing and Semantic Role Labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient Inference and Structured Learning for Semantic Role Labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online Projectivisation for Synchronous Parsing of Semantic and Syntactic Dependencies. In *In Proceedings of the International Joint Conference on Artificial Intelligence*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A Global Joint Model for Semantic Role Labeling. *Computational Linguistics*, 34(2).

David Vickrey and Daphne Koller. 2008. Applying Sentence Simplification to the CoNLL-2008 Shared Task. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Luke Zettlemoyer and Michael Collins. 2009. Learning Context-dependent Mappings from Sentences to Logical Form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*.

Hai Zhao and Chunyu Kit. 2008. Parsing Syntactic and Semantic Dependencies with Two Single-stage Maximum Entropy Models. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*.