Using Semantic Cues to Learn Syntax

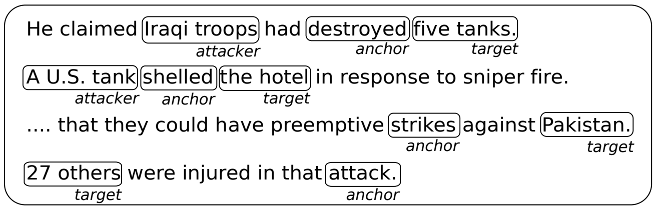
Tahira Naseem and Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{tahira, regina} @csail.mit.edu

Abstract

We present a method for dependency grammar induction that utilizes sparse annotations of semantic relations. This induction set-up is attractive because such annotations provide useful clues about the underlying syntactic structure, and they are readily available in many domains (e.g., info-boxes and HTML markup). Our method is based on the intuition that syntactic realizations of the same semantic predicate exhibit some degree of consistency. We incorporate this intuition in a directed graphical model that tightly links the syntactic and semantic structures. This design enables us to exploit syntactic regularities while still allowing for variations. Another strength of the model lies in its ability to capture non-local dependency relations. Our results demonstrate that even a small amount of semantic annotations greatly improves the accuracy of learned dependencies when tested on both in-domain and out-of-domain texts.[1]

Introduction

In this paper, we investigate the benefits of using partial semantic annotations for dependency grammar induction. While the connection between syntactic and semantic structures is commonly modeled in NLP, in most cases the goal is to improve semantic analysis using syntactic annotations. In this paper, we leverage this connection in the opposite direction. This formulation is inspired by studies in cognitive science that demonstrate that semantic constraints on what "makes sense" drive the process of syntactic acquisition (Piantadosi et al. 2008). From the practical viewpoint, the proposed set-up is attractive because partial semantic annotations are readily available in multiple domains. Examples include HTML markup (Spitkovsky, Jurafsky, and Alshawi 2010), Freebase tables (Mintz et al. 2009), and infoboxes. Moreover, while producing syntactic annotations requires significant amount of expertise (Bies et al. 1995), partial semantic annotations can be easier to elicit from a lay annotator. For instance a number of information extraction papers rely on Amazon Mechanical Turk to collect annotations (Wu and Weld 2010).


He claimed Iraqi troops had destroyed five tanks.

A U.S. tank shelled the hotel in response to sniper fire.

.... that they could have preemptive strikes against Pakistan.

27 others were injured in that attack.
attacker
anchor
target
attacker
anchor
target
anchor
target
target
anchor

Figure 1: Example sentences from the ACE corpus for *Conflict-Attack* event.

Our approach is based on the intuition that syntactic realizations of the same semantic relation exhibit some degree of consistency. Consider, for instance, the first two sentences from Figure 1. Both of them verbalize the relation *Conflict-Attack* using the same dependency structure. This similarity in structure provides a strong constraint for dependency grammar induction. For instance, we can require that a parser derives identical structures for all the instances of the same semantic relation. In practice, however, such a constraint is too strong — frequently, syntactic realizations of the same semantic relation can exhibit some variations. Examples of such variations are shown in Figure 1 where the relation is expressed using a noun phrase in sentence 3 and as a passive construction in sentence 4. Therefore, the key modeling challenge is to encourage syntactic consistency while still allowing for variations.

To achieve this effect, we propose a directed graphical model that simultaneously explains both the syntactic and semantic structure of a sentence. It first generates a dependency tree, and then conditioned on that, it generates semantic labels for the tree nodes. Our model posits that each syntactic structure has a different distribution over the set of semantic labels. This design implicitly encourages frequent reuse of the same syntactic structure per semantic relation to achieve a high probability derivation. At the same time, it allows several syntactic structures to favor the same semantic label. Since semantic relations can be expressed using arbitrarily long dependency paths, the model has to capture non-local syntactic patterns. As a result, the probability of the dependency tree cannot be decomposed into edge fac-

[1]The source code for the work presented in this paper is available at http://groups.csail.mit.edu/rbg/code/semdep/

tors, which makes inference complicated. We address this issue using a Metropolis-Hastings sampler.

We apply the proposed method to the task of dependency parsing for two datasets — the Wall Street Journal (Marcus, Santorini, and Marcinkiewicz 1993) and the ACE 2005 Corpus (Walker et al. 2006). As a source of semantic annotations we use relations developed for information extraction on the ACE corpus. The semantic annotations are sparse and noisy — only 17% of constituent boundaries are immediately apparent from the semantic labels, of which 20% are incorrect. These annotations are provided during training only, not at test time. Our results on both corpora demonstrate that the proposed method can effectively leverage semantic annotations despite their noise and sparsity. When compared to a version of the model without semantic information, we observe absolute performance gains of roughly 16 points on both in-domain and out-of-domain test corpora. We also demonstrate that improvement in learned syntactic structure is observed throughout the tree and is not limited to fragments expressing the semantic relations.

Related Work

Despite the magnitude of papers that leverage syntactic information in information extraction and semantic role labeling (Ge and Mooney 2009; Poon and Domingos 2009), research on the use of semantic constraints in grammar induction is surprisingly sparse. The set-up was first considered in the context of symbolic parsing, where manually constructed domain-specific grammar was used to reduce the ambiguity of syntactic derivations (Ghosh and Goddeau 1998). While the paper demonstrated the benefits of adding semantic information to syntactic parsing, the amount of hand-crafted knowledge it requires is prohibitive. More recently, Spitkovsky, Jurafsky, and Alshawi revisited this problem in the context of modern grammar induction. They demonstrated that using HTML mark-up as a source of semantic annotations brings moderate gains to dependency accuracy. They incorporate the mark-up information by treating HTML tags as constituent boundaries. This constituency information is translated into a variety of hard dependency constraints which are enforced during Viterbi-EM based learning. While we also treat semantic labels as constituent spans, we incorporate this constraint into the structure of our generative model. In addition, we also capture the syntactic relationship between semantic labels, which is not modeled by Spitkovsky, Jurafsky, and Alshawi.

Our work also relates to research on improving grammar induction using external knowledge sources. Our approach is closest to a method for grammar induction from partially-bracketed corpora (Pereira and Schabes 1992). Like our approach, this method biases the induction process to be consistent with partial annotations. The key difference, however, is that in our case the semantic annotations are noisy and are not at the same level of granularity as the target annotation (i.e., dependency trees). Therefore, they cannot be directly mapped to hard constraints on the induction process. This property distinguishes our work from recent approaches that incorporate declarative linguistic knowledge using expectation constraints on the posterior (Druck, Mann, and McCallum 2009; Naseem et al. 2010). For instance, in this framework it is not clear how to encode the preference that the dependency paths be consistent.

Model

The central hypothesis of this work is that knowledge about the semantic structure of a sentence can be helpful in learning its syntactic structure. In particular we capitalize on the fact that semantic concepts exhibit consistent patterns in their syntactic realizations. To leverage this idea, we propose a directed graphical model that generates semantic labels conditioned on syntactic structure. Such a model will encourage the selection of syntactic structures that have consistent patterns across different instances of the same semantic concept.

Semantic Representation

The model takes as input a set of sentences annotated with part-of-speech tags. Moreover, for each sentence, partial semantic annotations are available. To be specific, one or more events are annotated in each sentence. An event annotation comprises an *anchor* word that triggers the event and one or more *argument* spans. Each *argument* span is labeled with its role in the event. There are a fixed number of event types, and for each event type there are a fixed number of argument labels. For example for the event type *Conflict-Attack*, the argument labels are *Attacker*, *Target* and *Place*. Figure 1 shows example sentences with event annotations. Event annotations are available only during training. No semantic annotations are provided at test time.

Linguistic Intuition

The semantic annotations described above can constrain the syntactic structure in multiple ways. First, the syntactic relationship between an *anchor* word and its arguments is somewhat consistent across multiple instances of the same event type. For example, in the first two sentences in Figure 1, the *Attacker* argument is the subject of the *anchor* verb. However, strongly enforcing this consistency might actually hurt the quality of the learned syntactic analysis. While there is some regularity in the syntactic structure, the variations are also significant. For instance, in the 499 occurrences of the *Conflict-Attack* event in our data, we observe 265 unique syntactic structures connecting the *anchor* word with the *Target* argument. Moreover, the most frequent of these structures covers only 8% of the instances and another 33 distinct structures are needed to cover 50% of the instances[2]. Our model posits a different distribution over semantic labels for each syntactic structure. This allows multiple syntactic structures to favor the same semantic label.

Another clue about the syntactic structure is available in the form of argument labels: the argument spans form syntactic constituents in most cases. For dependency parsing this implies that there should be no outgoing dependency edges from an argument span. This holds true in our dataset roughly 80% of the time. We incorporate this constraint into

[2]These statistics are based on the dependency trees produced by a supervised parser.

our model by assuming that if a tree node corresponds to an argument label, all of its children will also be part of the same argument.

The final restriction that our model imposes on the syntactic structure is that a function word cannot be the head of a semantic argument. This is a reasonable assumption because function words by definition have minimal semantic content. In our data, this assumption is always true.

Generative Process

Our model first generates the syntactic structure of a sentence i.e. the dependency tree. Next, for every event type, an *anchor* node is selected randomly from all the nodes in the tree. Finally, an *argument* label for every node in the tree is generated conditioned on the dependency path from the *anchor* node. Below we describe the generative process in more detail. A summary is given in Table 1.

Generating Dependency Trees The process for generating a dependency tree starts with generating a root node. Then child nodes are generated first towards the right of the root node until a stop node is generated, and then towards the left until a stop node is generated. This process is then repeated for every child node.

Each node comprises a part-of-speech tag drawn from a multinomial $\theta_{p,d,v}$ over the set of part-of-speech symbols, where p is the parent part-of-speech tag, d is the direction of the child node with respect to the parent node i.e. right or left and v is the valence of the parent. Valence encodes how many children have been generated by the parent before generating the current child. It can take one of the three values: 0, 1 or 2. A value of 2 indicates that the parent already has two or more children.

This component of the model is similar to the Dependency Model with Valence (DMV) of (Klein and Manning 2004). The only difference is that our model encodes more detailed valence information.

Generating Events After generating the dependency tree, event labels are drawn conditioned on the tree structure. For every event type e first an anchor node is selected uniformly from all the nodes in the tree plus a null node. The null node accounts for the case where the event does not occur in the sentence. If the selected node is the null node, the generation process stops, otherwise we proceed to generate the argument labels.

To generate argument labels, tree nodes are traversed in a top down fashion starting from the root. Whenever a content word node is visited[3], an argument label is generated for the node. This argument label is drawn from a multinomial $\phi_{e,path}$. The support of this distribution is over all allowable labels for the arguments of event type e plus a special *not-argument* label. If the label generated at a node is the *not-argument* label, the process continues recursively on the child nodes. Otherwise, the process stops for that node, and all the nodes dominated by the argument node become a part of the argument. Forcing all children of an argument node to have the same argument label allows the model to benefit from the constituency boundaries indicated by the argument annotations.

The argument generation multinomial ϕ is indexed by event type and path – i.e. we have one such multinomial for each event type and each unique path from an anchor node to a potential argument node. This path is specified by concatenating the part-of-speech tags along the path, joined by arrows indicating the direction of each node with respect to its parent node. Figure 2 shows example paths from the anchor node to two potential argument nodes.

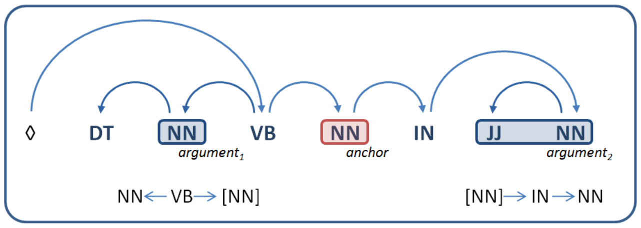


Figure 2: A dependency tree with anchor and argument annotations. Path from $anchor$ node to $argument_1$ (left) and $argument_2$ (right) are also given. Square brackets indicate the *anchor* nodes.

Generating Parameters We have two sets of model parameters: syntactic parameters i.e. the child tag generation multinomials $\theta_{\mathbf{p,d,v}}$ and semantic parameters i.e. the event argument generation multinomials $\phi_{\mathbf{e,path}}$. Both these sets of multinomials are drawn from conjugate Dirichlet priors.

The child tag multinomial distributions $\theta_{\mathbf{p,d,v}}$ are drawn from a symmetric Dirichlet with hyperparameter θ_0. The event argument distributions $\phi_{\mathbf{e,path}}$ are drawn from a non-symmetric Dirichlet prior where the hyperparameters for all argument labels are set to ϕ_0 except for the hyperparameter for the *not-argument* label, which is set to a value ϕ_0' proportional to the length of the $path$. This prior bias incorporates the intuition that the dependency path from anchor to argument should be short. A long path to a node strongly indicates that the node does not have an argument relationship with the anchor.

Inference

During training we want to estimate the posterior probability of latent dependency trees given the observed part-of-speech tags and event annotations. Directly estimating this posterior requires computing marginals over all possible settings of the model parameters for every possible set of dependency trees for the whole corpus. This makes exact inference intractable. Instead, we employ Gibbs sampling to approximate the model posterior. This allows us to sample one latent variable at a time, conditioning on the current values of all other latent and observed variables. The only variables that we sample during inference are the dependency trees; model parameters θ and ϕ are marginalized out.

The inference procedure starts by assigning an initial dependency tree to every sentence. During each iteration, a dependency tree is sampled for each sentence from the con-

[3]Content word nodes can be identified based on their part-of-speech tags which have already been generated; e.g. Nouns, Verbs, Adjectives and Adverbs are the major classes of content words.

```
For each POS tag $t$, each valence $v$ and direction $d$:
    Draw child tag multinomial $\theta_{t,d,v} \sim \mathrm{Dir}(\theta_0)$.
For each event type $e$ and dependency path $p$:
    Draw argument multinomial $\phi_{e,p} \sim \mathrm{Dir}(\phi_0)$.
For each sentence $s_i$:
    Draw dependency tree $T_i$ as follows:
        For each parent tag $t$ with valence $v$ in direction $d$:
            Draw child tag $t' \sim \mathrm{Mult}(\theta_{t,d,v})$.
    For each event type $e$:
        Select anchor node $c_e$ uniformly from nodes in $T_i$.
        For each node $n \in T_i$ with path $p$ from $c_e$:
            Draw argument label $a \sim \mathrm{Mult}(\phi_{e,p})$
```

Table 1: The generative process for model parameters, dependency parses and event labels. In the above Dir and Mult refer respectively to Dirichlet distribution and multinomial distribution.

ditional probability distribution over trees given the current samples for all other sentences. Conditional probability of the dependency tree T_i for the i^{th} sentence is given by,

$$P(T_i|x_i, e_i, T_{-i}, x_{-i}, e_{-i}, \theta_0, \phi_0) \propto P(T_i|x_i, T_{-i}, x_{-i}, \theta_0) P(e_i|T_i, T_{-i}, e_{-i}, \phi_0), \quad (1)$$

where x_i is the tag sequence and e_i is a set of event annotations for the i^{th} sentence. The notation m_{-i} represents all variables m except m_i. The first term on the right side corresponds to generation of the tree structure, while the second term corresponds to generation of the semantic event labels given the tree structure. We can further expand the first term as follows:

$$P(T_i|x_i, T_{-i}, x_{-i}, \theta_0) = \prod_{j=1}^{n} P(x_i^j | p_{x_i^j}, v, d, T_{-i}, x_{-i}, \theta_0)$$

where n is the sentence length, $p_{x_i^j}$ is the parent tag of tag x_i^j according to dependency tree T_i, v is the valence of the parent node, and d is the direction of the tag x_i^j with respect to its parent according to dependency tree T_i. Thus the probability of a tree is a product of the probabilities of its nodes. The child generation multinomial parameters θ are marginalized out. This marginalization has a closed form due to Dirichlet-multinomial conjugacy. Each term in the product can be computed using following equation:

$$P(x|p_x, v, d, T_{-i}, x_{-i}, \theta_0) = \frac{count(x, p_x, v, d) + \theta_0}{count(p_x, v, d) + k\theta_0} \quad (2)$$

where k is the total number of unique tags. $count(x, p_x, v, d)$ is the number of times tag x has been generated by tag p_x in direction d with valence v. Similarly, $count(p_x, v, d)$ is number of times parent tag p_x generated any tag in direction d with valence v.

The second term on the right side of Formula 1, which is responsible for the generation of semantic event labels, expands as follows:

$$P(e_i|T_i, T_{-i}, e_{-i}, \phi_0) \propto \prod_{j=1}^{z} P(e_i^j|T_i, T_{-i}, e_{-i}, \phi_0) \quad (3)$$

where z is the total number of event types.

Each event annotation e comprises an anchor word and a set of argument labels one for each word in the sentence. For an event with event type t_e, let c_e represent the tree node corresponding to the anchor word and let $a_e(n)$ represent the argument label for node n in the dependency tree T_i. Hence, each term on the right of Formula 3 expands into:

$$\begin{aligned} &P(e|T_i, T_{-i}, e_{-i}, \phi_0) \\ &= P(c_e|t_e, T_i, T_{-i}, e_{-i}, \phi_0) P(a_e|c_e, t_e, T_i, T_{-i}, e_{-i}, \phi_0) \\ &= \frac{1}{|T_i \cup null|} \prod_{n \in T_i} P(a_e(n)|path(n, c_e, T_i), t_e, \phi_0) \end{aligned}$$

where $path$, as described in model section, is the dependency path from anchor node c_e to current node n according to dependency tree T_i. Each term in the product on the right side has a closed form similar to Equation 2.

Directly sampling dependency trees from Distribution 1 is not possible, because the term related to event generation involves conditioning on non-local dependency paths. As a result the probability does not factor along the edges of the dependency tree, which makes it intractable to construct the marginalized probability tables (i.e. the inside tables) required for sampling trees. To circumvent this problem we use a Metropolis-Hastings (MH) sampler. Rather than directly sampling from $P(T_i|T_{-i}, \mathbf{x}, \mathbf{e}, \theta_0, \phi_0)$ we instead sample from a tractable proposal distribution $Q(T_i|T_{-i}, \mathbf{x}, \theta_0)$ and then accept the sample with the following probability:

$$\min\left\{1, \frac{P(T|T_{-i}, \mathbf{x}, \mathbf{e}, \theta_0, \phi_0) Q(T'|T_{-i}, \mathbf{x}, \theta_0)}{P(T'|T_{-i}, \mathbf{x}, \mathbf{e}, \theta_0, \phi_0) Q(T|T_{-i}, \mathbf{x}, \theta_0)}\right\}$$

where T is the newly sampled tree for the i^{th} sentence and T' is the previous sample for the same sentence. The proposal distribution that we use is similar to the true model distribution given in Formula 1, except it does not have the term involving events generation. In other words the proposal distribution assumes that the generation process stops after generating dependency trees. We selected this proposal distribution because it is close to the actual model distribution while still being tractable.

To sample a dependency tree from the proposal distribution, we use the standard tree sampling algorithm (Johnson, Griffiths, and Goldwater 2007). We first construct the inside probability table for the sentence. Then starting from the top we sample one dependency decision at time based on the inside probabilities below it, thus marginalizing over the probabilities of all possible decisions below.

Experimental Setup

Data We train our model on the English portion of the ACE 2005 Multilingual Training Corpus (Walker et al. 2006). The data sources for this corpus include newswire, broadcast news, broadcast conversations, weblog and telephone conversations. The ACE corpus is annotated with entities, relations and events. We use only the event annotations in our experiments. The event annotations are restricted to 18 event types. In any given sentence the only events annotated are those that fall into one of these pre-specified types.

We extract only those sentences from the corpus that contain at least one event annotation. Following standard practice, punctuation markers and special symbols are removed. Any sentences longer than 50 words, after removing punctuation, are filtered out. This process leaves us with a total of 2442 sentences with 3122 event annotations. On average, there are 1.3 event annotations per sentence.

As explained in the model section an event annotation comprises an anchor and one or more argument annotations. In our data, the average number of arguments per event is 2.3. There are roughly 4.2 semantic labels per sentence, including both anchor and argument labels. 43.4 percent of the words are covered by at least one semantic label. However most of this coverage comes from very long argument spans, where no information is available about the internal structure of the argument. If we count only the headword of each argument span, the word coverage is only 19.2%.

The ACE corpus does not have part-of-speech annotations which are required by our model. We obtain these annotations using the Stanford Tagger version 3.0 (Toutanova and Manning 2000).

Training and Testing We test our model on two datasets: the ACE corpus and the WSJ corpus. We split the 2442 sentences extracted from the ACE corpus into 2342 training sentences and 100 test sentences. These test sentences are selected randomly. We manually annotate these sentences with dependency trees, following the same linguistic conventions that are used by the Penn converter (Johansson and Nugues 2007). Event annotations are ignored during testing. From WSJ, we use section 23 for testing. For comparison with previous work, we test on both the sentences of length 10 or less (WSJ10) and sentences of all lengths (WSJ). Dependency trees for this corpus are produced using the Penn converter (Johansson and Nugues 2007).

Initialization and Decoding The initial dependency trees are set to the Viterbi output of the standard DMV model trained on our training data using the EM algorithm.

In all the experiments, θ_0, the hyperparameter for the syntactic child generation multinomial, is set to 0.25 and ϕ_0, the hyperparameter for the semantic argument generation multinomials, is set to 1. The hyperparameter ϕ'_0 corresponding to the *not-argument* label, is set to $10 \times$ path-length, where path-length is the number of dependency edges along the path. All hyperparameters remain fixed during training. We train the model for 5000 sampling iterations.

At test time, Viterbi decoding is performed using only the syntactic parameters of the model. These parameters are set to the MLE estimates based on the counts from the final training sample with hyperparameters used for smoothing.

Baselines We compare our model against several baselines. First we construct a baseline by using only the syntactic component of our model. We train this baseline on the training data without making use of the event annotations. Since this baseline is very close to our main model, a comparison with it should highlight the effect of using semantic annotations. We use the same initialization and parameter settings for the baseline that we used in our main experiments.

We also compare against the results of Spitkovsky, Jurafsky, and Alshawi. Their setup is the closest to ours since they use HTML mark-up to guide grammar induction.

In addition, we compare against the best available unsupervised parsing results on WSJ (Blunsom and Cohn 2010) and WSJ10 (Headden III, Johnson, and McClosky 2009).

Finally, following the setup of (Spitkovsky, Jurafsky, and Alshawi 2010), we compare with the supervised MLE version of the syntactic component of our model. The parameters are set to the MLE estimates based on the gold dependencies from the WSJ corpus, with hyperparameters used for smoothing. The performance of this model can be viewed as an upper bound for our main model, since our main model also uses only the syntactic component at test time.

Results

Table 2 shows the result for our model along with the results for various baselines. The results on the in-domain ACE corpus show that the availability of semantic annotations significantly improves the performance, cutting the gap between the supervised MLE results and the unsupervised baseline results by about 62%. Similar trends can be observed in the results on the out-of-domain Wall Street Journal corpus.

		ACE50	WSJ	WSJ10
1	Baseline	46.6	43.2	56.2
2	Full Model	**63.4**	**59.4**	**70.2**
3	Blunsom et al. (2010)	-	55.7	67.7
4	Headden et al. (2009)	-	-	68.8
5	Spitkovsky et al. (2010)	-	50.4	69.3
6	Supervised MLE	72.9	70.5	82.2

Table 2: Directed dependency accuracy of our model on in-domain ACE corpus and out-of-domain WSJ corpus. The baseline corresponds to the syntax-only component of our model. Rows 3 and 4 show the results of the state-of-the-art unsupervised dependency parsers. Row 5, (Spitkovsky, Jurafsky, and Alshawi 2010), corresponds to a model trained using HTML mark-up. The last row shows the results based on the supervised MLE version of our model.

For the ACE dataset we also evaluated the performance separately on the words that are covered by the argument labels. We compute the percentage of these words whose head is predicted correctly by the model. Our model gives an accuracy of 62.5% on these words compared to the overall accuracy of 63.4%. This result shows that we in fact learn a better parser rather than just improving on those structures involved in semantic annotations.

Our model also outperforms the state-of-the-art unsupervised models on the WSJ corpus by a significant margin, despite the fact that the underlying syntactic component of our model is much simpler than the models used by Headden III, Johnson, and McClosky and Blunsom and Cohn. We believe this implies that replacing the syntactic component of our model with a more sophisticated one has the potential of further improving the results.

Ablation Experiments As explained earlier in the model section, our model exploits event annotations by assuming that, 1) the arguments form constituents, 2) the head of a semantic argument is always a content word and 3) there are consistent patterns in dependency paths from the anchor node to the argument node across multiple occurrences of the same event type. To analyze the relative contribution of each of these assumptions, we perform ablation experiments by dropping them one at a time.

Table 3 shows the results of the ablation experiments. The impact of dropping the first two constraints is much more drastic than the path constraint. This is partly due to the fact that the first two constraints are much more consistently followed in the gold data than the path constraint. Another reason why removing the path information has little effect on performance is that the presence of the head constraint indirectly enforces path consistency.

	ACE50	WSJ	WSJ10
Full Model	**63.4**	**59.4**	**70.2**
No argument brackets	43.5	41.3	48.5
No head constraint	50.1	46.8	57.4
No path	61.9	58.4	69.0

Table 3: Results of the ablation experiments.

We also performed an experiment where we only use the bracketing constraint. This approximates the setup used in (Pereira and Schabes 1992). This setup gives an accuracy of 48.3% on the ACE corpus, which is higher than the baseline (46.6%) but significantly worse than the full model performance (63.4%). The reason for this low performance might be that the brackets available in the form of argument labels, are very sparse.

Conclusions

We presented a method for dependency grammar induction that utilizes annotations of semantic relations. We demonstrated that even incomplete annotations of semantic relations provide useful clues for learning syntactic structure.

Acknowledgements

The authors acknowledge the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0172 and the U. S. Army Research Laboratory and the U. S. Army Research Office under contract no. W911NF-10-1-0533. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Bies, A.; Ferguson, M.; Katz, K.; MacIntyre, R.; Tredinnick, V.; Kim, G.; Marcinkiewicz, M.; and Schasberger, B. 1995. Bracketing Guidelines for Treebank II Style Penn Treebank project. *University of Pennsylvania.*

Blunsom, P., and Cohn, T. 2010. Unsupervised Induction of Tree Substitution Grammars for Dependency Parsing. In *Proceedings of EMNLP*, 1204–1213.

Druck, G.; Mann, G.; and McCallum, A. 2009. Semi-supervised Learning of Dependency Parsers using Generalized Expectation Criteria. In *Proceedings of ACL/IJCNLP*, 360–368.

Ge, R., and Mooney, R. 2009. Learning a Compositional Semantic Parser Using an Existing Syntactic Parser. In *Proceedings of ACL/IJCNLP*, 611–619.

Ghosh, D., and Goddeau, D. 1998. Automatic Grammar Induction from Semantic Parsing. In *Fifth International Conference on Spoken Language Processing*.

Headden III, W. P.; Johnson, M.; and McClosky, D. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of NAACL/HLT*, 101–109.

Johansson, R., and Nugues, P. 2007. Extended Constituent-to-dependency Conversion for English. In *Proceedings of NODALIDA*, 105–112.

Johnson, M.; Griffiths, T.; and Goldwater, S. 2007. Bayesian Inference for PCFGs via Markov Chain Monte Carlo. In *Proceedings of NAACL-HLT*, 139–146.

Klein, D., and Manning, C. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*, 478–485.

Marcus, M. P.; Santorini, B.; and Marcinkiewicz, M. A. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics* 19(2):313–330.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL/IJCNLP*, 1003–1011.

Naseem, T.; Chen, H.; Barzilay, R.; and Johnson, M. 2010. Using Universal Linguistic Knowledge to Guide Grammar Induction. In *Proceedings of EMNLP*.

Pereira, F., and Schabes, Y. 1992. Inside-Outside Reestimation from Partially Bracketed Corpora. In *Proceedings of ACL*, 128–135.

Piantadosi, S.; Goodman, N.; Ellis, B.; and Tenenbaum, J. 2008. A Bayesian model of the acquisition of compositional semantics. In *Proceedings of the Thirtieth Annual Conference of the Cognitive Science Society*.

Poon, H., and Domingos, P. 2009. Unsupervised Semantic Parsing. In *Proceedings of EMNLP*, 1–10.

Spitkovsky, V. I.; Jurafsky, D.; and Alshawi, H. 2010. Profiting from Mark-Up: Hyper-Text Annotations for Guided Parsing. In *Proceedings of ACL*, 1278–1287.

Toutanova, K., and Manning, C. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-speech Tagger. In *Proceedings of EMNLP/VLC*, 63–70.

Walker, C.; Strassel, S.; Medero, J.; and Maeda, K. 2006. Ace 2005 Multilingual Training Corpus. *Linguistic Data Consortium, Philadelphia.*

Wu, F., and Weld, D. 2010. Open Information Extraction Using Wikipedia. In *Proceedings of ACL.*