Online Graph Planarisation for Synchronous Parsing of Semantic and Syntactic Dependencies

Ivan Titov University of Illinois at U-C titov@illinois.edu

James Henderson, Paola Merlo, Gabriele Musillo University of Geneva {James.Henderson,Paola.Merlo,Gabriele.Musillo}@unige.ch

Abstract

This paper investigates a generative history-based parsing model that synchronises the derivation of non-planar graphs representing semantic dependencies with the derivation of dependency trees representing syntactic structures. To process non-planarity online, the semantic transition-based parser uses a new technique to dynamically reorder nodes during the derivation. While the synchronised derivations allow different structures to be built for the semantic non-planar graphs and syntactic dependency trees, useful statistical dependencies between these structures are modeled using latent variables. The resulting synchronous parser achieves competitive performance on the CoNLL-2008 shared task, achieving relative error reduction of 12% in semantic F score over previously proposed synchronous models that cannot process non-planarity online.

1 Introduction

Significant advances in natural language processing applications will require the development of systems that exhibit some shallow representation of meaning. Parsing techniques have successfully addressed semantic problems such as recovering the logical form of a sentence for information extraction [Wong and Mooney, 2007]. Many current methods for shallow semantic parsing follow syntactic parsing in focusing on parsing models for labelled directed graphs that form trees. While the space of tree structures is sufficiently constrained to apply standard parsing algorithms, it is not expressive enough to represent many semantic phenomena, such as dependencies between the predicates in a sentence and their respective arguments. Lexicalised unification-based grammars have explicitly modelled such linguistic facts with directed graphs that are not trees.

In this paper, we develop a generative model for the labelled directed graphs recently used to represent syntactic and semantic dependencies. Figure 1 illustrates the kind of structures that are studied here. Following the CoNLL-2008 shared task formalism [Surdeanu *et al.*, 2008], we assume a dependency formalism for syntax, as well as a dependency

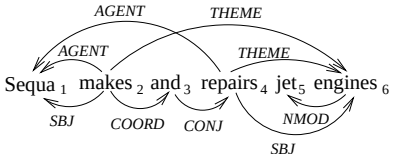

Figure 1: A non-planar semantic dependency graph labelled with semantic roles paired with a planar syntactic dependency tree labeled with grammatical relations.

formalism for the relation between a predicate and its arguments: Directed arcs in the dependency graph represent the semantic relations between the predicates and the arguments, and labels associated with the arcs encode semantic roles. As can be observed in Figure 1, semantic dependency structures are very different from syntactic dependency structures. Syntactic dependencies form trees, and only 7.6% of sentences contain crossing arcs in their syntactic structures, in the data provided by the CoNLL-2008 shared task. In contrast, semantic dependency structures are in general not trees, since they do not form a connected graph and some nodes have more than one parent. In the CoNLL-2008 data, only 22% of sentences have semantic structures which can be treated as trees. Also, 43% of these sentences have semantic structures that contain crossing arcs. These fundamental differences motivate the development of new techniques specifically for handling semantic dependency structures.

Following the recent approach of Henderson *et al.* [2008], we capture the different nature of these two linguistic levels by two synchronised transition-based systems that separately derive the syntactic structure and the semantic structure. Differently from Henderson *et al.* [2008], however, we do not attempt to extend the standard methods for un-crossing arcs (called planarisation) to the semantic structure.

For the semantic structure, instead, we augment the transition system with a new operation, that we call *Swap*, which disentangles crossing arcs online. We demonstrate that this parsing algorithm is sufficiently powerful to parse 99% of the semantic graphs in the training set of the CoNLL-2008 shared task. Also, the resulting model achieves an improvement of about 3% in F_1 score on labelled semantic dependencies over the previous synchronous model of Henderson *et al.* [2008].

Our probabilistic model is based on Incremental Sigmoid Belief Networks (ISBNs), a recently proposed latent variable model for syntactic structured prediction, which has shown

very good behaviour for both constituency [Titov and Henderson, 2007b] and dependency parsing [Titov and Henderson, 2007c]. The use of latent variables enables this architecture to be extended to learning a synchronous parse of syntax and semantics [Henderson *et al.*, 2008]. This model maximises the joint probability of the syntactic and semantic dependencies and thereby enforces that the output structure be globally coherent, but the use of synchronous parsing allows it to maintain separate structures for the syntax and semantics.

The best model we have trained achieves 81.8% macro-average F_1 performance for the joint task, which would correspond to the fifth position in the ranking of systems participated in the CoNLL-2008 shared task, and first in the ranking of systems that learn the syntax and semantics jointly. Importantly, ours is also the best system which does not use either model combination or reranking. It is therefore simpler, and a good candidate for use as a component in an ensemble.

In what follows, we introduce the online planarisation technique in section 2; we briefly review the synchronous parsing method and learning architecture we use in sections 3 and 4; we report and discuss the experimental results in section 5; we relate this work to existing work, and draw some conclusions, in sections 6 and 7.

2 Non-Planar Parsing

The differences between syntactic and semantic structures make it difficult to apply syntactic dependency parsing techniques to semantic dependency parsing. Because they are not trees, it is impossible to apply dependency parsing algorithms based on Minimum Spanning Tree algorithms (e.g. [McDonald *et al.*, 2005]) directly to semantic dependency structures. It is fairly straightforward to adapt transition-based parsing algorithms such as [Nivre *et al.*, 2006] to such structures [Henderson *et al.*, 2008; Sagae and Tsujii, 2008], but these algorithms inherit the constraint from their tree-parsing counterparts that the structures be planar. Planarity requires that the graph can be drawn in the semi-plane above the sentence without any two arcs crossing, and without changing the order of words.[1]

As will be discussed in section 6, there have been multiple approaches to transition-based non-planar parsing for dependency trees. The most common have been approaches which first transform a non-planar tree into a planar tree with extended labels, and then apply planar parsing [Nivre and Nilsson, 2005]. We use such an approach [Henderson *et al.*, 2008] as our baseline. Another approach is to extend the parsing model itself so that it can parse arbitrary non-planar structures [Attardi, 2006]. In this paper we adopt a simplified version of this approach, where we introduce a single new action. Although the resulting parser is not powerful enough to parse all non-planar structures, this single action can handle the vast majority of non-planar structures which occur in the data.

2.1 Non-Planar Parsing using Swapping

For parsing non-planar graphs, we introduce an action *Swap*, which swaps the top two elements on the parser's stack. We add this action to the transition-based parsing algorithm for planar graphs proposed in Henderson *et al.* [2008], which is based on Nivre's parsing algorithm [Nivre *et al.*, 2006].

In the Henderson *et al.* [2008] planar parsing algorithm, the state of the parser is defined by the current stack S, the queue I of remaining input words, and the partial labeled dependency structure constructed by previous parser actions. The parser starts with an empty stack S and terminates when it reaches a configuration with an empty input queue I. The algorithm uses four types of actions:

1. The action *Left-Arc*$_r$ adds a dependency arc from the next input word w_j to the word w_i on top of the stack and selects the label r for the relation between w_i and w_j.
2. The action *Right-Arc*$_r$ adds an arc from the word w_i on top of the stack to the next input word w_j and selects the label r for the relation between w_i and w_j.
3. The action *Reduce* pops the word w_i from the stack.
4. The action *Shift*$_{w_j,s}$ shifts the word w_j from the queue to the stack. It also marks the next input word as a predicate with sense s or declares that it is not a predicate.

In this paper, we propose the addition of the *Swap* action:

5. The action *Swap* swaps the two words at the top of the stack.

The *Swap* action is inspired by the planarisation algorithm described in Hajičová *et al.* [2004], where non-planar trees are transformed into planar ones by recursively rearranging their sub-trees to find a linear order of the words for which the tree is planar (also see the discussion of Nivre [2008] in section 6). For trees, such an order is guaranteed to exist, but for semantic graphs this is not the case. For example, there is no such order for the semantic dependency graph in the top half of Figure 1. Rather than first sorting and then parsing a planar structure, the *Swap* action allows us to reorder words online during the parse. This allows words to be processed in different orders during different portions of the parse, so some arcs can be specified using one ordering, then other arcs can be specified using another ordering.

This style of parsing algorithm allows the same structure to be parsed multiple ways. Rather than trying to sum over all possible ways to derive a given structure, which would be computationally expensive, models are trained to produce parses in a canonical order. We have tried two canonical parsing orders. Both orders only use swapping when it is needed to uncross arcs, but they differ in when the swapping is done.

The first canonical parsing order we use in this paper tries to perform *Swap* actions at positions where they are predictable, and therefore can be easily learned. This order only uses the *Swap* action as a last resort, when no other action is possible. With this ordering the *Swap* action is used when the word under the top of the stack needs to be attached to the front of the queue, which is a decision we would hope to be able to learn. Unfortunately, this ordering is not completely general: in the CoNLL-2008 data, 2.8% fewer semantic structures are parsable with this ordering than are possible with the *Swap* action in general. For example, the structure in Figure 2 cannot be parsed with this ordering, even though

[1] Some parsing algorithms require *projectivity*, this is a stronger requirement that disallows not only crossing arcs but also edges covering the root node [Nivre and Nilsson, 2005].

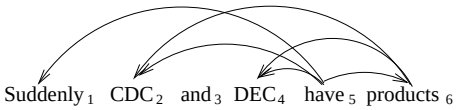

Figure 2: An example structure cannot be parsed with the last-resort algorithm though there exists a derivation: *Shift(1)*, *Shift(2)*, *Swap(1,2)*, *Shift(3)*, *Reduce(3)*, *Shift(4)*, *Left-Arc(4,5)*, *Swap(1,4)*, *Left-Arc(1,5)*, *Reduce(1)*, *Swap(2,4)*, *Left-Arc(2,5)*, *Shift(5)*, *Right-Arc(5,6)*, *Reduce(5)*, *Left-Arc(2,6)*, *Reduce(2)*, *Left-Arc(4,6)*, *Reduce(4)*, *Shift(6)*.

there exist a sequence of actions which derives it. We will call this canonical parse ordering the *last-resort* algorithm.

To define a canonical ordering which is guaranteed to find a derivation if one exists, we need to make use of swapping preemptively to uncross future arcs. This ordering follows a standard planar parsing order until there are no other actions possible except for *Swap* and *Shift*. At this point it computes the ordered list of positions of words in the queue to which the word w_i on the top of the stack should be connected in the remaining part of the parse. A similar list should be computed for word w_j under the top of the stack. These two lists are compared using lexicographical order and if word w_j's list precedes word w_i's list, then they must be swapped. Otherwise, the *Shift* action is performed. In Figure 2, after the action *Shift(2)*, the list of future arcs for word CDC_2 on the top of the stack is equal to $\{5,6\}$ and the list for word $Suddenly_1$ under the top of the stack is $\{5\}$. $\{5\}$ precedes $\{5,6\}$ in the lexicographical order, therefore *Swap* should be performed. We call this algorithm the *exhaustive* algorithm.

Theorem 1. *If a graph is parsable with the set of operations defined above then the exhaustive algorithm is guaranteed to find a derivation.*

Proof sketch. Space constraints do not allow us to present the proof, so we explain only the intuition behind the algorithm, which is relatively straightforward to expand into a formal proof. All the attachment actions are performed between a word on the top of the stack and a word in the queue. Therefore, when deciding on the order of two elements on the top of the stack we should prefer to place on top the word which will be attached sooner (A). If the next attachment for both words happens with the same queue then we should prefer either to move up the word which can be reduced from the stack immediately after the attachment (B) or to move up the word which will participate in the subsequent attachment earlier than the other word (C). Note, that all these tests (A-C) are implicitly embedded in the test of the lexicographical order between the lists of their future connections. □

Both these algorithms extend existing canonical orders with a decision for when to swap. In our experiments, we apply these extensions to the arc-eager late-reduce strategy, where we keep words in the stack even after they are connected to all their children and parents in the graphs. Such 'processed' words are removed from the stack only when they prevent other operations, such as attaching words under 'processed' words on the stack or swapping words separated by one or more 'processed' words. In preliminary experiments, we found that this late-reduce strategy leads to improved performance, as observed previously [Nivre *et al.*, 2006].[2]

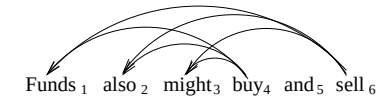

Figure 3: A non-planar semantic dependency graph that cannot be parsed with swapping.

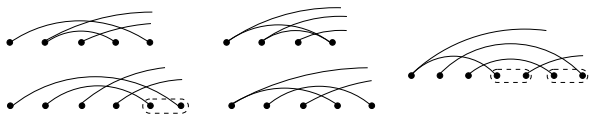

Figure 4: Configurations of arcs which cannot be parsed with the *Swap* action.

2.2 The Structures Parsable with Swapping

The class of structures parsable with swapping covers a surprising proportion of sentences. In our experiments on the CoNLL-2008 shared task dataset [Surdeanu *et al.*, 2008], introducing the *Swap* action was sufficient to parse the semantic dependency structures of 38,842 out of 39,279 training sentences (99%). Of these, 16,993 sentences required a *Swap* to be parsed (43%). In these sentences, the *Swap* action was used 31,110 times for the exhaustive algorithm, and 55,071 times for the last-resort algorithm, which is 0.15 swaps per arc and 0.27 swaps per arc, respectively.

From a linguistic point of view, among many linguistic structures which this parsing algorithm can handle without any construction dependent-operations, one of the frequent ones is coordination. The algorithm can process coordination of two conjuncts sharing a common argument or being arguments of a common predicate, for instance, *Sequa makes and repairs jet engines*, as well as similar structures with three verb conjuncts and two arguments, for instance *Sequa makes, repairs and sells jet engines*.[3]

In general, the *Swap* action can parse any isolated pair of crossing arcs. However, not all the configuration where a single arc crosses more than one other arc can be parsed. A frequent example of an unparsable structure which involves 3 arguments attached to 2 predicates is presented in Figure 3.

Theorem 2. *A graph cannot be parsed with the defined set of parsing operations iff the graph contains at least one of the subgraphs presented in Figure 4, the unspecified arc endpoints can be anywhere strictly-following those specified, and circled pairs of endpoints can either be a single word or two distinct words.*[4]

[2] Nivre *et al.* [2006] used a late-reduce strategy for all the languages in the CoNLL-2005 shared task. See http://w3.msi.vxu.se/users/jha/conllx/ for details.

[3] The structure of a typical non-planar semantic graph involving coordination is illustrated in Figure 1, whose derivation is the sequence of actions *Shift(1)*, *Right-Arc(1,2)*, *Shift(2)*, *Swap(1,2)*, *Shift(3)*, *Reduce(3)*, *Right-Arc(1,4)*, *Shift(4)*, *Shift(5)*, *Reduce(5)*, *Left-Arc(4,6)*, *Reduce(4)*, *Reduce(1)*, *Left-Arc(2,6)*, *Reduce(2)*, *Shift(6)*.

[4] Note that the directionality of the arc is unimportant.

Proof sketch. Again, due to space considerations, we are not able to provide a detailed proof here, but the proof strategy is the following. If a graph is unparsable then there exists a derivation state where two words A and B on the top of the stack have their rightmost attachment after the next attachment of some word C deeper in the stack. Then all the possible linear word orders for A, B and C are considered. For each such an order all the arc configurations which lead to the described final derivation state are then derived. Note that according to Theorem 1 it is sufficient to consider only derivations defined by the exhaustive algorithm. □

3 Synchronous derivations

We synchronize syntactic and semantic derivations using the model of Henderson *et al.* [2008]. The derivations for syntactic dependency trees are the same as those specified above for semantic dependencies, but there is no *Swap* action and the other actions are more constrained in when they can apply.[5]

Let T_d be a syntactic dependency tree with derivation $D_d^1, ..., D_d^{m_d}$, and T_s be a semantic dependency graph with derivation $D_s^1, ..., D_s^{m_s}$. To define derivations for the joint structure T_d, T_s, we specify that the two derivations are synchronised at every word.

We divide the two derivations into the chunks between shifting each word onto the stack, $c_d^t = D_d^{b_d^t}, ..., D_d^{e_d^t}$ and $c_s^t = D_s^{b_s^t}, ..., D_s^{e_s^t}$, where $D_d^{b_d^t-1} = D_s^{b_s^t-1} = \textit{Shift}_{t-1}$ and $D_d^{e_d^t+1} = D_s^{e_s^t+1} = \textit{Shift}_t$. Then the actions of the synchronous derivations consist of quadruples $C^t = (c_d^t, \textit{Switch}, c_s^t, \textit{Shift}_t)$, where *Switch* means switching from syntactic to semantic mode. This gives us the following joint probability model, where n is the number of words:

$$P(T_d, T_s) = P(C^1, ..., C^n) = \prod_t P(C^t | C^1, ..., C^{t-1}).$$

The probability of each synchronous derivation chunk C^t is the product of four factors, related to the syntactic level, the semantic level and the two synchronising steps:

$$P(C^t|C^1, \ldots, C^{t-1}) =$$
$$P(c_d^t|C^1, ..., C^{t-1}) P(\textit{Switch}|c_d^t, C^1, ..., C^{t-1}) \times$$
$$P(c_s^t|\textit{Switch}, c_d^t, C^1, ..., C^{t-1}) P(\textit{Shift}_t|c_d^t, c_s^t, C^1, ..., C^{t-1}).$$

These synchronous derivations $C^1, \ldots, C^n$ only require a single input queue, since the *Shift* actions are synchronised, but they require two separate stacks, one for the syntactic derivation and one for the semantic derivation.

The probability of c_d^t is decomposed into derivation action D^i probabilities, and likewise for c_s^t:

$$P(c_d^t|C^1, ..., C^{t-1}) = \prod_i P(D_d^i | D_d^{b_d^t}, ..., D_d^{i-1}, C^1, ..., C^{t-1}).$$

[5]The amount of non-planarity in syntax for this dataset is very small and, therefore, the choice of the parsing strategy for non-planar syntactic dependencies cannot seriously affect the performance of our method. We used the standard HEAD pre-/post-processing method of Nivre and Nilsson [2005] for syntax.

4 The Learning Architecture

The synchronous derivations described above are modelled with an Incremental Sigmoid Belief Network (ISBN) [Titov and Henderson, 2007a]. They have previously been applied to constituency parsing [Titov and Henderson, 2007b], dependency parsing [Titov and Henderson, 2007c], and synchronous syntactic-semantic parsing [Henderson *et al.*, 2008]. ISBNs are dynamic Bayesian Networks which use vectors of latent state variables to represent features of the parsing history relevant to the future decisions. Our ISBN model distinguishes two types of latent states: syntactic states, when syntactic decisions are considered, and semantic states, when semantic decision are considered. These latent variable vectors are conditioned on variables from previous states via a pattern of edges determined by the previous decisions. For these we adopt a set of edges previous proposed in Henderson *et al.* [2008], namely those for their "large" model, which includes latent-to-latent connections both from syntax states to semantics states and vice versa.

Word	Semantic step features			
	LEX	POS	DEP	SENSE
Next	+	+	+	+
Top	+	+	+	+
Top - 1	+	+	+	
LDep Next			+	
Head Top/Top-1	+	+	+	
Head Next	+	+	+	
RDep Top/Top-1			+	
LDep Top/Top-1			+	
LSib Top/Top-1		+	+	
LSib Next		+	+	
RSib Top/Top-1		+	+	
RSib Next		+	+	

Table 1: Features for semantic states. Columns identify feature types, rows identify words (with respect to the queue and the semantic stack), and a + identifies which features are used for which words. Next= front of input queue; Top= top of stack; Top-1= element below top of stack; R/LDep= rightmost/leftmost dependent; R/LSib= right/left sibling.

The latent variable vectors are also conditioned on a set of observable features of the derivation history. For these features, we extended the features proposed in Henderson *et al.* [2008]. The set of observable features for syntactic states is left unchanged, and the set of observable features for semantic states given in Table 1 is expanded to allow better handling of the non-planar structures in semantics. Most importantly, all the features of the top of the stack are now also included for the word just under the top of the stack.

5 Experiments and Discussion

We train and evaluate our models on data provided for the CoNLL-2008 shared task on joint learning of syntactic and semantic dependencies. The data is derived by merging a dependency transformation of the Penn Treebank with Propbank and Nombank [Surdeanu *et al.*, 2008]. An illustrative example of the kind of labelled structures that we need to parse was given in Figure 1. More details and references on

TECHNIQUE	CONLL MEASURES			CROSSING PAIRS		
	Synt	SRL	M	Semantics		
	LAS	F_1	F_1	P	R	F_1
Last resort	86.6	76.2	81.5	61.5	25.6	36.1
Exhaustive	86.8	76.0	81.4	59.7	23.5	33.8
HEAD	86.7	73.3	80.1	78.6	2.2	4.2
Planar	85.9	72.8	79.4	und	0	und

Table 2: Scores on the development set; Und= undefined; SRL= semantic graph; M F_1= Macro F_1

MODEL	CONLL MEASURES			CROSSING PAIRS		
	Synt	SRL	M	Semantics		
	LAS	F_1	F_1	P	R	F_1
Johansson	89.3	81.6	85.5	67.0	44.5	53.5
Ciaramita	87.4	78.0	82.7	59.9	34.2	43.5
Che	86.7	78.5	82.7	56.9	32.4	41.3
Zhao	87.7	76.7	82.2	58.5	36.1	44.6
This Paper	87.5	76.1	81.8	62.1	29.4	39.9
Henderson	87.6	73.1	80.5	72.6	1.7	3.3
Lluis	85.8	70.3	78.1	53.8	19.2	28.3

Table 3: Scores on the test set; SRL= semantic graph; M F_1= Macro F_1

the data, the conversion of the Penn Treebank format to dependencies, and on the experimental set-up are given in Surdeanu *et al.* [2008].

We compare several experiments in which we manipulate different variants of online planarisation techniques for the semantic component of the model. The models are illustrated in Table 2. We compare both the last resort (first line) and the exhaustive strategy (second line) to two baselines. The first baseline (third line) uses Nivre's HEAD label propagation technique to planarise the syntactic tree, extended to semantic graphs following Henderson *et al.* [2008]. The second baseline is an even simpler baseline that only allows planar graphs, and therefore fails on non-planar graphs (fourth line). In training, if a model fails to parse an entire sentence, it is still trained on the partial derivation.[6]

In our experiments, we use the measures of performance used in the CoNLL-2008 shared task, typical of dependency parsing and semantic role labelling. Syntactic performance is measured by percentage of correct labelled attachments (LAS in the tables) and semantic performance is indicated by the F-measure on precision and recall on semantic arcs (indicated as SRL measures in the tables). These two components are then averaged in a score called Macro F_1. To evaluate directly the impact of the *Swap* action on crossing arcs, we also calculate precision, recall and F-measure on pairs of crossing arcs. In the case of multiple crossings, a link can be a member of more than one pair.

The results of these experiments are shown in Table 2. The results are clear. If we look at the left panel of Table 2 (CoNLL Measures), we see that the last resort strategy perform the best, and that both online planarisation techniques outperform the extension of Nivre's technique to semantic graphs (third line) and the simplistic baseline. Clearly, the improvement is due to better recall on the crossing arcs, as shown by the right-hand panel.

These experiments were run on the development set. The best performing model (LAST RESORT) was then tested on the test set and compared to some other models that participated in the CoNLL-2008 shared task. The models were chosen among the 20 participating systems either because they had better results or because they learnt the two representations jointly. Results of these experiments on the test sets are summarised in Table 3. The method reported here is an improvement on the best performing single systems (Henderson). Specifically, while the already competitive syntactic performance is not significantly degraded, we report an improvement of 3% on the semantic graphs. This score approaches those of the best systems. As the righthand panel on crossing arcs indicates, this improvement is due to better recall on crossing arcs. Also, importantly, this model is one of the few that does joint learning, with the best results in that category. Four systems, however, can report better performance than our system. The best performing system learns the two representations separately, with a pipeline of state-of-the-art systems, and then reranks the joint representation in a final step [Johansson and Nugues, 2008]. Similarly, Che *et al.* [2008] also implement a pipeline consisting of state-of-the-art components where the final inference stage is performed using Integer Linear Programming to ensure global coherence of the output. The other two better performing systems use ensemble learning techniques [Ciaramita *et al.*, 2008; Zhao and Kit, 2008]. If we take into account the fact that ours is the best single-system, joint learner, we can confirm that joint learning is a promising technique, but that on this task it does not outperform reranking or ensemble techniques. The system's architecture is, however, simpler.

Other joint models do not perform as well as our system. In Lluis and Marquez [2008] a fully joint model is developed, that learns the syntactic and semantic dependencies together as a single structure. This differentiates their approach from our model, which learns two separate structures, one for syntax and one for semantics, and relies on latent variables to represent the interdependencies between them. It is not clear whether it is this difference in the way the models are parameterised or the difference in the estimation techniques used that gives us better performance, but we believe it is the former.

6 Related Work

Approaches to dealing with non-planar graphs belong to two conceptual groups: those that manipulate the graph, either by pre-processing or by post-processing, and those that adapt the algorithm to deal with non-planarity.

Among the approaches that, like ours, devise an algorithm to deal with non-planarity, already Yngve [1960] proposed a limited manipulation of registers to handle discontinuous constituents, which guaranteed that parsing/generation could be performed with a stack of very limited depth.

An approach to non-planar parsing which is more similar to ours has been proposed in Attardi [2006]. Attardi's dependency parsing algorithm adds six new actions, which allows this algorithm to parse any type of non-planar tree. Our *Swap*

[6] All variants use the same set of features and interconnections, latent variable vectors of size 80, and a word frequency cut-off of 5. The data is parsed with a beam search algorithm described in Henderson *et al.* [2008] with a beam of 20.

action is related to Attardi's actions *Left2* and *Right2*, which create dependency arcs between the second element on the stack and the front of the input queue. In this algorithm, every attachment to an element below the top of the stack requires the use of one of the new actions, whose frequency is much lower than the normal attachment actions, and therefore harder to learn. This contrasts with the *Swap* action, which handles reordering with a single action, and the normal attachment operations are used to make all attachments to the reordered word. Though much simpler, this single action can handle the vast majority of crossing arcs which occur in the data.

In a recently published paper, Nivre [2008] presents the formal properties of a swap action for dependency grammars that enables parsing of non-planar structures. The formal specifications of this action are different from the specifications of the action proposed here. Nivre's action can swap terminals repeatedly and move them down to an arbitrary point into the stack. This *Swap* action can potentially generate word orders that cannot be produced by only swapping the two topmost elements in the stack. However, when defining the oracle parsing order for training, Nivre [2008] assumes that the dependency structure can be planarised by changing the order of words. This is not true for many of the semantic dependency graphs, because they are not trees.

The most common approach to dealing with non-planar structures is to transform crossing arcs into non-crossing arcs with augmented labels [Nivre and Nilsson, 2005]. One drawback of this approach is that it leads to a leaky probability model, in that structures with augmented labels that do not correspond to any tree receive non-zero probabilities. When parsing with such a model, the only computationally feasible search consists in finding the most likely augmented structure and remove inconsistent components of the dependency graph [Nivre *et al.*, 2006; Titov and Henderson, 2007c]. But this practically-motivated method is not equivalent to a statistically motivated – but computationally infeasible – search for the most probable consistent structure. Moreover, learning these graphs is hard because of the sparseness of the augmented labels.

A chart-parsing algorithm targeting a subclass of non-planar structures was very recently proposed in Kuhlmann and Satta [2009]. However, they have not constructed and evaluated statistical models based on their formalism.

Other solutions apply data-driven transforms to the output of a strictly planar (projective) dependency parser, as in corrective modelling [Hall and Novak, 2005] and approximate non-projective parsing [McDonald and Pereira, 2006].

7 Conclusions

In this paper, we report on a online technique to parse non-planar structures that handles many of the graphs used to represent predicate-argument semantics. This technique is embedded in a synchronous dependency parser for syntax and semantics that learns these two representations jointly. In the future we will study the applicability of our online planarisation technique to syntactic parsing of languages with highly non-planar syntactic representations.

Acknowledgements

The authors thank Dan Roth and the reviewers for helpful comments. This work was partly funded by European Community FP7 grant 216594 (CLASSiC, www.classic-project.org), Swiss NSF grant 114044, Swiss NSF fellowships PBGE2-117146 and PBGE22-119276, NSF grant SoD-HCER-0613885, DARPA funding under the Bootstrap Learning Program.

References

G. Attardi. Experiments with a multilanguage non-projective dependency parser. In *Proc. CoNLL*, 2006.

W. Che, Zh. Li, Y. Hu, Y. Li, B. Qin, T. Liu, and S. Li. A cascaded syntactic and semantic dependency parsing system. In *Proc. of CoNLL*, 2008.

M. Ciaramita, G. Attardi, F. DellOrletta, and M. Surdeanu. DESRL: A linear-time semantic role labeling system. In *Proc. of CoNLL*, 2008.

E. Hajičová, J. Havelka, P. Sgall, K. Veselá, and D. Zeman. Issues of Projectivity in the Prague Dependency Treebank. (81), 2004.

K. Hall and V. Novak. Corrective modeling for non-projective dependency parsing. In *Proc. of the Int. Workshop on Parsing Technology (IWPT'05)*, 2005.

J. Henderson, P. Merlo, G. Musillo, and I. Titov. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proc. of CoNLL*, 2008.

R. Johansson and P. Nugues. Dependency-based syntactic–semantic analysis with propbank and nombank. In *Proc. of CoNLL*, 2008

M. Kuhlmann and G. Satta. Treebank grammar techniques for non-projective dependency parsing. In *Proc. EACL*, 2009.

X. Lluis and L. Marquez. A joint model for parsing syntactic and semantic dependencies. In *Proc. of CoNLL*, 2008.

R. McDonald and F. Pereira. Online learning of approximate dependency parsing algorithms. In *EACL*, 2006.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of EMNLP*, 2005.

J. Nivre and J. Nilsson. Pseudo-projective dependency parsing. In *CoNLL* 2005.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, and S. Marinov. Pseudo-projective dependency parsing with support vector machines. In *CoNLL*, 2006.

J. Nivre. Sorting out dependency parsing. In *Proc. of GoTAL*, 2008.

K. Sagae and J. Tsujii. Shift-reduce dependency DAG parsing. In *Proc. of COLING*, 2008.

M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL*, 2008.

I. Titov and J. Henderson. Incremental Bayesian networks for structure prediction. In *ICML*, 2007.

I. Titov and J. Henderson. Constituent parsing with incremental sigmoid belief networks. In *Proc. ACL*, 2007.

I. Titov and J. Henderson. A latent variable model for generative dependency parsing. In *Proc. of the Int. Conf. on Parsing Technologies (IWPT'07)*, 2007.

Y. W. Wong and R. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proc. ACL*, 2007.

V. H. Yngve. A model and an hypothesis for language structure. *American Philosophical Society*, 104(5), 1960.

H. Zhao and C. Kit. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Proc. of CoNLL 2008*, Manchester, UK, 2008.