Evaluation of Two-level Dependency Representations of Argument Structure in Long-Distance Dependencies

Paola Merlo
Linguistics Department
University of Geneva
1211 Geneva 4
Switzerland
Paola.Merlo@unige.ch

Abstract

Full recovery of argument structure information for question answering or information extraction requires that parsers can analyse long-distance dependencies. Previous work on statistical dependency parsing has used post-processing or additional training data to tackle this complex problem. We evaluate an alternative approach to recovering long-distance dependencies. This approach uses a two-level parsing model to recover both grammatical dependencies, such as subject and object, and full argument structure. We show that this two-level approach is competitive, while also providing useful semantic role information.

1 Introduction

One of the main motivations for adopting dependency representations in the parsing and computational linguistics community is their direct expression of the lexical-semantic properties of words and their relations. Argument structure is the representation of the argument taking properties of a predicate. It represents those semantic properties of a predicate that are expressed grammatically. It is usually defined as the specification of the arity of the predicate, its grammatical functions and the substantive labels of the arguments in the structure, what are usually called thematic or semantic roles. For example the argument structure of the verb *hit* comprises the specification that *hit* is a transitive verb and that it takes an AGENT subject and a THEME object.

Constructions involving long-distance dependencies (LDDs) — such as questions, or relative clauses — are the stress test of the ability to represent argument structure, because in these constructions argument structure information is not directly reflected in the surface order of the sentence. Despite the complexity of their representation, Rimell et al. (2009) report that these constructions cover roughly ten percent of the data in a corpus such as the PennTreebank, and therefore cannot be ignored. LDDs are illustrated in Figure 1. Representing argument structure in long-distance dependency constructions, thus, requires special mechanisms to deal with the divergence between the argument taking properties of the verb and the surface order of the sentence. The most frequently used ways to encode long-distance dependencies is either by a copy mechanism, shown in Figure 1, or by turning the tree into a directed graph, shown in Figure 2. [1]

Many current statistical dependency parsers fail to represent many long-distance dependencies and their related argument structure directly, often because the relevant information, such as traces, has been stripped from the training data. For example, most current statistical parsers do not represent directly the links drawn below the sentences in Figure 2. Moreover, there is no attempt in these representations, to encode the full argument structure directly, as the semantic role labels are usually inferred from their correlation with the grammatical function labels, but not explicitly represented. The argument structure of the verb *spread* in the first sentence in Figure 2 comprises a THEME subject in the intransitive form of the verb. This argument structure must be inferred indirectly from the graph: first the long-distance *nsubj* relation

[1] Recall that the red arcs shown in the figures are for expository purposes only, current representations do not show these direct links for long-distance dependencies.

Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015), pages 221–230,
Uppsala, Sweden, August 24–26 2015.

must be inferred from a sequence of links typical of subject extraction from an embedded clause. Moreover, the notion that verbs like *spread* take THEME subjects in some, but not all cases, is not represented, and therefore the argument structure cannot be, strictly speaking, fully recovered.

These parsers can recover the long-distance dependency only through a post-processing step, which recovers the information about predicate-argument relation and the grammatical function. The semantic role label is usually not recovered even in post-processing.

We investigate here, then, the hypothesis whether current two-level syntactic-semantic parsers can fill in for the missing information, and recover the long-distance and argument structure information during parsing without need for post-processing and without loss in performance. If this were possible, we would be able to produce long-distance dependencies with more direct and perspicuous representations, and also fill in some of the semantic information currently missing from argument structure representations.

It is important to recall that the reason why predicate-argument structure is considered central for NLP applications hinges on the assumption that what needs recovering is the lexical semantics content. For example, it is likely that for information extraction, it is more useful to know which are the manner, temporal and location arguments than to know an underspecified adverbial modifier label.

In the rest of the paper, then, we will first contrast the one-level representation of long-distance dependencies to a two-level representation, where grammatical functions and argument structure are both explicitly represented. We will then briefly recall a recently proposed two-level parsing model (Henderson et al., 2013), and then present the main contribution of the paper: the evaluation of parsing models that parse these two-level syntactic-semantic dependencies on long-distance dependencies. We also compare the results to other statistical dependency parsers, investigate the usefulness and informativeness of the extracted information, discuss and conclude.

2 Single-level and Two-level Encoding of LDDs

As discussed in the introduction, traditional linguistic encodings of LDDs are integrated in the parse tree, either as co-indexed "traces", such as in the Penn Treebank, as illustrated in Figure 1, or as arcs as in a dependency representation. In practice, current statistical parsers do not encode LDD directly, as illustrated in Figure 2, and leave it to post-processing procedures to recover the LDD relation (Johnson, 2002; Nivre et al., 2010). These approaches exploit the very strong constraints that govern long-distance relations syntactically, and ignore the full or partial recovery of the semantic roles entirely.

Consider, for example, the representations for subject embeddings (first tree) and object reduced relatives (second tree) in Figure 2. This figure illustrates the Stanford dependency representation that was used in Rimmel et al. (2009), and Nivre et al. (2010), indicating below the sentence the long distance dependency that needs to be recovered, but that is not in the representation. The first tree encodes the subject relation between *ac-*

(1) Questions

$What_i$ did William hit_i with his arrow?

(2) Relative clauses

This is the $apple_i$ that William hit_i with his arrow.

Figure 1: LDDs and their coindexed antecedent-trace representation.

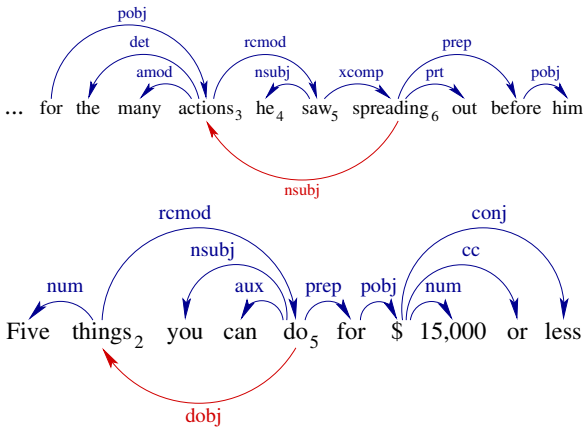

Figure 2: LDDs represented as a syntactic dependency tree labeled with grammatical relations. Recall that the LDD encoded in the arcs under the sentence are the LDD that must be recovered. They are shown for expository purpose and they are not usually part of the syntactic tree.

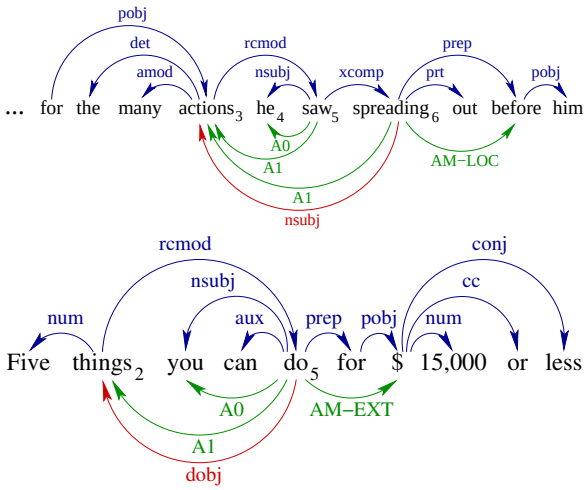

Figure 3: LDDs represented as a syntactic dependency tree above the sentence (in blue) and argument structure labels under the sentence (in green). The label A0 stands for AGENT and A1 stands for THEME. The prefix AM indicates a modifier argument. Recall that the LDDs encoded in the arcs under the sentence (in red) are the LDDs that must be recovered. They are shown for expository purpose and they are neither part of the syntactic tree nor of the semantic graph.

tions and *spreading* as a sequence of two arcs *rcmod(actions, saw)* and *xcomp(saw, spreading)*. This sequence indicates a dependency relation in the opposite direction from the one needed to correctly recover the argument structure of the verb *spread*, and does not explicitly indicate the grammatical function,SUBJECT, nor the semantic role relation, THEME. The label *rcmod* is the same label used to indicate the relationship between *do* and *things* in the second sentence, but in this case the relation is an object relation, so the distinction between subject-oriented and object-oriented relative clauses is encoded very indirectly. This kind of encoding of argument structure and long-distance dependency is indirect and potentially lacking in perspicuity.

In a dependency formalism, two-level representations have been proposed to represent the syntactic and argument structures of a sentence in terms of dependencies. Consider the representations in Figure 3. The syntactic representation is the same as in the previous figures, but LDDs and argument structures are represented directly. For example, the verb *saw* has two arguments, an AGENT and a THEME, while the verb *spread* has a long-distance dependency with the word *actions*, which is its THEME subject.[2] The verb *do* in the second sentence has a long-distance THEME object. Therefore, the overall complex graph that represents both the syntax and the underlying argument structure of the sentences comprises two half graphs, sharing all vertices, the words. They are indicated by the blue and green arcs, respectively, in Figure 3.

These representations factor the syntactic parse tree information from the argument structure information and provide, overall, more labelling information. The parse tree is needed to provide a connected graph, to provide information about constituency/dependency relations for grammatical correctness (agreement, for example, is triggered in environments defined by grammatical functions, and not by semantic relations) and grammatical functions. Argument structures are represented separately, for each predicate in the sentence and give explicit labels to the arguments. While these labels are correlated to the grammatical functions, it is a well-established fact that they are not coextensive, for instance not all subjects are Agents as shown in Figure 3, and therefore are not redundant.

From a linguistic point of view, these representations are related to many grammar formalisms that invoke the need to represent both grammatical functional level and argument structure level, such as tectogrammatical dependency representations (Hajic, 1998), or early versions of transformational grammar.

From a graph-theoretic and parsing point of view, the complete graph of both the syntax and the semantics of the sentences is composed of two half graphs, which share all their vertices, namely the words. Internally, these two half graphs exhibit different properties. The syntactic graph is a single connected tree. The semantic graph is a forest of one-level treelets, one for each proposition, which may be disconnected and may share children. In both graphs, it is not generally appropriate to assume independence across the different treelets in the structure. In the semantic graph, linguistic evidence that propositions are not independent of each other comes from constructions such

[2]This sentence also exemplifies the well-known fact, referred to in the introduction, that the mapping from grammatical function to the semantic roles useful for interpretation is not simple: the subject is not an AGENT, the most frequent mapping, but a THEME.

as coordinations where some of the arguments are shared and semantically parallel. Arcs in the semantic graph do not correspond one-to-one to arcs in the syntactic graph, indicating that a rather flexible framework is needed to capture the correlations between graphs. The challenge, then, arises in developing models of these two-level representations. These models must find an effective way of communicating the necessary information between the syntax and the argument structure representation.

From the practical point of view of existing resources, one version of these representations results from the merging of widely used and carefully annotated linguistic resources, PennTreebank (Marcus et al., 1993) and PropBank (Palmer et al., 2005). They are PennTreebank-derived dependency representations that have been stripped of long-distance dependencies, and merged with PropBank encoding of argument structures. But PropBank encodings are often based on the trace-enriched PennTreeBank representations as a starting point. Hence, these representations encode all LDDs, enriched with substantive semantic role labels, according to the PropBank labelling scheme.[3] They could also be constructed from other resources, for example by augmenting the current Universal dependency annotation scheme with extra semantic annotations (de Marneffe et al., 2014).

3 Parsing Two-level Representations

Developing models to learn these two-level analyses of syntax and argument structure raises several interesting questions regarding the design of the interface between the syntactic and the argument structure representations and how to learn these complex representations (Merlo and Musillo, 2008; Surdeanu et al., 2008).[4]

A model that can parse these two level-dependencies is proposed in Henderson et al. (2013) and we adopt it here without modifications. We choose this model for our evaluation of long-distance dependencies as the best performing among those approaches that have attempted to model jointly the relationship between argument structure and surface syntax (Lluís and Màrquez, 2008; Surdeanu et al., 2008) and developments of this model have shown good performance on several languages (Gesmundo et al., 2009), without any language-specific tailoring. These results suggest that this model can capture abstract linguistic regularities in a single parsing architecture.[5] We describe this model here very briefly. For more detail on the parser and the model, we refer the interested reader to Henderson et al. (2013) and references therein.

The crucial intuitions behind the two-level approach is that the parsing mechanism must correlate the two half-graphs, but allow them to be constructed separately as they have very different properties. The derivations for both syntactic dependency trees are based on a standard transition-based, shift-reduce style parser (Nivre et al., 2006). The derivations for argument structure dependency graphs use virtually the same set of actions, but are augmented with a *Swap* action, that swaps the two words at the top of the stack. The *Swap* action is inspired by the planarisation algorithm described in Hajicova 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.

The probability model to determine which action to pursue is a joint generative model of syntactic and argument structure dependencies. The two dependency structures are specified as the synchronised sequences of actions for a shift-reduce parser that operates on two different stacks. By synchronising parsing for both the syntactic and the argument structure representations, a probabilistic model is learnt which maximises the joint probability of the syntactic and semantic dependencies and thereby guarantees that the output structure is globally coherent, while at the same time building the two structures separately. The probabilistic estimation is based on Incremental Sigmoid Belief Networks (ISBNs). The use of latent variables allows ISBNs to induce their fea-

[3] These representations are the same, in practice, as the encoding used in some recent shared tasks (CoNLL 2008 and CoNLL 2009 (Surdeanu et al., 2008; Hajič et al., 2009)) for syntactic-semantic dependencies.

[4] Joint syntactic-semantic dependency parsing was the theme of two CoNLL shared tasks. CoNLL 2008 explored syntactic-semantic parsing for English, CoNLL 2009 extended the task to several languages. Only four truly joint models were developed, and most of the multi-lingual models were fine-tuned specifically for each language.

[5] The version of the parser, the one we use, described in Henderson et al. (2013), has a syntactic labelled accuracy of 87.5%, a semantic role F-score of 76.1%, and a syntactic-semantic F-score of 81.8%, using the data and evaluation measures of the CoNLL 2008 shared task.

(3) Each must match Wisman's pie with the fragment that they carry with him.

(4) Five things you can do for 15,000 dollars or less.

(5) They will remain on a lower-priority list that includes 17 other countries.

(6) How he felt ready for the many actions he saw spreading out before him.

(7) What you see are self-help projects.

(8) What effect does a prism have on light?

(9) The men were at first puzzled then angered by the aimless tacking.

Figure 4: Sentences exemplifying the different constructions involving LDDs, used in the test set developed by Rimell et al. (2009).

tures automatically.

4 Experiments

In this section we assess how well the two-level parser performs on constructions involving long-distance dependencies. In so doing, we verify that these two-level models of syntactic and argument structure representations can be learnt even in difficult cases, while also producing an output that is richer than what statistical parsers usually produce. To confirm this statement, we expect to see that the syntactic dependency parsing performance is not degraded, compared to more standard statistical parsing architectures on long-distance dependencies, while also producing semantic role labels on these difficult constructions.

4.1 The Test Data

To test the performance on LDDs, we use the test suites developed by Rimell et al. (2009) for English. They comprise 560 test sentences, 80 for each type of construction. Half of them are extracted from the Penn Treebank, half of them from the Brown corpus, balanced across construction types. None of these sentences is included in the training set of the parser. These sentences cover seven types of long-distance relations, illustrated in Figure 4: object extraction from relative clauses (ORC) in (3) or from reduced relative clauses (ORed) in (4), subject extraction from relative clauses (SRC) in (5) or from an embedded clause (SEmb) in (6), free relatives (Free) in (7), object-oriented questions (OQ) in (8), and right node raising constructions (RNR) in (9).

Compared to the other statistical dependency parsers, questions (OQ) are not well represented in our training data, since they do not include the additional QB data (Nivre et al., 2010) used to improve the performance of MSTParser and MaltParser.

4.2 Parsing set up

Like the dependency parser in Nivre et al. (2010), the parser was not trained on the same data or tree representations as those used in the test data. The parser is trained on the data 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 3. Training and development data follow the usual partition as sections 02-21, 24 of the Penn Treebank. More details and references on the data, and the conversion of the Penn Treebank format to dependencies are given in Surdeanu et al. (2008).

Like for standard statistical and dependency parsers, the syntactic representation used by the two-level parser has been stripped of all traces. The predicates of the argument structures and their locations are not provided at testing, unlike some of the CONLL shared tasks.

Unlike Nivre et al. (2010), we did not use an external part-of-speech tagger to annotate the data of the development set. To minimize preprocessing of the data, we choose to have part-of-speech tagging as an internal part of the parsing model, which therefore, takes raw input.

In order for our results to be comparable to those reported in previous evaluations (Rimell et al., 2009; Nivre et al., 2010), we ran the parser "out of the box" directly on the test sentences, without using the development sentences to fine-tune. We were able to parse all the sentences in the test suites without any adjustments to the parser.[6]

[6] According to Rimell et al. (2009) only the C&C parser required some little adjustments to parse all sentences in the test suite. Evaluation results without these adjustments are not reported.

4.3 Evaluation Methodology

Like in previous papers (Rimell et al., 2009; Nivre et al., 2010), we evaluate the parser on its ability to recover LDDs. Two evaluations were done. The first one was semi-automatic, performed with a modified version of the evaluation script developed in Rimell et al. (2009). An independent manual evaluation was also performed.

A dependency is considered correctly recovered if a dependency in the gold data is found in the output. A dependency is a triple comprising three items: the nodes connected by the arc in the graph and the label of the arc. In principle, a dependency is considered correct if all three elements of the triple are correct. However, in this evaluation the representations vary across models and exact matches would not allow a fair assessment. Both previous evaluation exercises (Rimell et al., 2009; Nivre et al., 2010) suggest some avenues to relax the matching conditions, and define equivalence classes of representations.

4.3.1 Equivalence classes of arcs

To relax the requirement of exact match on the definition of arc, a set of equivalence classes between single arcs and paths connecting two nodes indirectly is precisely defined in the post-processing scheme of Nivre et al. (2010), which applies to the Stanford labelling scheme. In Nivre et al. (2010), the encoding of long-distance dependencies in a dependency parser is categorised as simple, complex, and indirect. In the simple case, the LDD coincides with an arc in a tree. In the complex case, the LDD is represented by a path of arcs. In the indirect case, the dependency is not directly encoded in a path in the tree, but it must be inferred from a larger portion of the tree using heuristics. The two last cases require post-processing of the tree. In Rimell et al. (2009), two dependencies are considered equivalent if they differ only in their definition of what counts as head. For example, in some dependency schemes the preposition is the head of a prepositional phrase, while in others it is the noun.

We develop a definition of equivalence classes of arcs inspired by both these approaches. Following Nivre et al. (2010), we define a long-distance dependency as simple or complex. In the simple case, the LDD coincides with an arc in a tree. A complex dependency is defined as a path of at most two simple dependencies. Unlike single-level statistical parsers, our two-level representation could create more than one path to connect two nodes, since two nodes could be connected both by a syntactic arc and by a semantic arc. Following Rimell et al. (2009), we define which path of two arcs is considered correct by allowing some flexibility in the definition of the head in very specific predefined cases, such as prepositional phrases. The head can be either the word in the position indicated in the gold annotation, or its parent. This definition applies, for example, to extraction from prepositional phrases which in our case are related to the semantic head, while in Rimell et al.'s scheme they are connected to the preposition. This relaxed definition is triggered in 31 cases of semantic matches and 40 cases of syntactic matches, over a total of 398 matches.

The evaluation script was also augmented with a construction-specific rule to capture complex dependencies with *be*-constructions. Sentence (10) is an example of a *be*-construction, where the gold dependency in (10a) corresponds to a path of two dependencies in (10b). The latter consists of the subject dependency between the copula *is*, the head, and its subject *childhood*, and the predicative dependency between the head *is* and the predicative *what*. For a complex dependency of this type to be counted correct, the end points of the path have to match the endpoints of the long-distance dependency in the gold and the labels have to be exactly as indicated, *sbj* and *prd*. This specific rule adds seven correct cases to the total.

(10) That is what childhood is , he told himself .

a. nsubj what 2 childhood 3
b. sbj is 1 childhood 3
   prd is 1 what 2

4.4 Equivalence classes of labels

The evaluation in Rimell et al. (2009) is largely done manually, and equivalences are decided by the authors. Different labelling schemes are considered correct, as long as they can make the distinction between subject, object, indirect object and adjunct modifier.

We establish a correspondence of labels. In our two-level representation, labels are the grammatical functions of the syntactic dependencies, and the semantic role labels, taken from PropBank. [7]

[7] The PropBank annotation was developed based on the deep structure representations of the PennTreebank and Levin

Core arguments	
nsubj	A0, A1, SBJ
obj,dobj, pobj	OBJ, A1, PMOD
passive subj	A1
obj2	A1
Other labels	
advmod	LOC,TMP,MNR
amod	MNR,NMOD
aux	MOD,VC
nn	NAME, DEP
partmod	MOD

Figure 5: Gold data and two-level output label equivalences.

Our equivalences might depend only on the labels or on the labels in the context of the sentence type. For example, the subject of a passive is an A1, that is a THEME. In some cases, direct inspection of the predicate was necessary: A1 corresponds to *subjects* for some verbs even in the active voice. A simple rule was applied to decide what verbs can exhibit an A1 subject, based on PropBank's framesets: If the frameset allowed A1 as a subject, in the appropriate sense of the verb, then the correspondence was accepted. This decision rule applied to 33 cases (the (nsubj, A1) cell in Table 2). The label equivalences are given in detail in Figure 5: the grammatical function labels of the gold data are shown on the left and labels of the two-level parser are shown on the right. The confusion matrix by labels is provided in Table 2.

Manual evaluation The evaluation was also done manually by a judge, a trained linguist, who had not developed the initial script. We used a visualisation tool (Tred) (Pajas and Štěpánek, 2008), adapted to our output, to facilitate the inspection of the two-level representations and avoid mistakes.

In the manual evaluation, a dependency is correctly recovered if an arc and its syntactic/semantic label (see Figure 4) are correct.

Three different constructions need to be mentioned, because they have special chracteristics that had to be taken into account: coordination, right node raising and small clauses.

A dependency may be found directly, as a single arc, or by coordination. Regarding coordination, we follow the Stanford scheme, according to which an argument or adjunct must be attached to the first conjunct to indicate that it belongs to both conjuncts.

Right node raising is too difficult to evaluate automatically. In Rimell et al. (2009)'s definition, right node raising is represented by two arcs. It is considered correctly recovered if one of the arcs was correct and the other was found either directly or by coordination. We evaluate right node raising by hand, in the same way: either the dependency was found directly or by coordination, either in the syntax or in the argument structure.

Small clauses are rare, complex dependencies that were evaluated by hand. Sentence (11) is an example of a small clause construction, where the *nsubj* dependency of the gold data (11a) corresponds to two dependencies (11b): one between the head *called* and its object/theme *horses*, and one between *called* and the object predicative *Dogs*. We found only five cases of this construction. However, these five dependencies do make a difference, because they all appear in SEmb, which has a low percent recall, as shown in Table 1.

(11) The sound rose on the other side of the hills , vanished and rose again and he could imagine the mad , disheveled hoofs of the Appaloosas , horses the white men once had called the Dogs of Hell .

a. nsubj Dogs 36 horses 28

b. obj called 34 horses 28
   oprd called 34 Dogs 36

5 Results and Discussion

Automatic and manual results (percent recall) are shown in Table 1, where we compare our results to the relevant ones of those reported in previous evaluations (Rimell et al., 2009; Nivre et al., 2010; Nguyen et al., 2012).[8] These papers compare several statistical parsers. Some parsers like Nguyen, the C&C parser (Clark and Curran, 2007) and Enju (Miyao and Tsujii, 2005) are based on rich grammatical formalisms, and others others are representative of statistical dependency parsers (MST, MALT, (McDonald, 2006; Nivre et al., 2006)).

(1993)'s semantic propositions of alternating verbs. PropBank propositions have been shown to be closely related to grammatical functions (Merlo and van der Plas, 2009). So we can assume that grammatical functions can also be inferred from PropBank relations in most cases.

[8] All these evaluations, like ours, can report only recall, because of the nature of the output of the parsers, which do not explicitly label a dependency with a dedicated long–distance label.

	ORC	ORed	SRC	Free	OQ	RNR	SEmb	Total
Nguyen	53	69	68	69	57	26	39	56
C&C	59	63	80	73	28	49	22	54
Enju	47	66	82	76	32	47	33	54
This paper	36/35	55/48	44/57	72/73	18/15	63	18/22	48/48
MST	34	47	79	66	14	46	38	46
Malt	41	51	84	70	16	40	24	46

Table 1: Our percent recall results, construction by construction, automatic and manual (A/M), compared to some of the results reported in Rimell et al. (2009) and Nivre et al. (2010). Abbreviations are explained in subsection 4.1. Right node raising was evaluated only manually.

	dobj	nsubj	pobj	prep
A0	0/6	37/39	0/2	
A1	146/146	12/33	6/6	3/3
A2	2/2			3/3
OBJ	16/16		1/1	
SBJ	0/7	10/10	0/4	
PMOD	5/5	0/1	13/13	2/2
TOT	167/186	75/87	20/26	25/25

Table 2: Labelled error confusion matrix of most frequent labels. Cells indicate correct labelling/total labelling. The first three rows show the results for semantic labels, and the last three rows show results for syntactic dependency labels. For reasons of space only labels with at least five occurrences are shown. The table also does not show the following perfect matches: LOC: advmod 5/5; TMP: advmod 5/5; A1: nsubjpass 14/14; NMOD: amod 7/7.

These last two parsers constitute the relevant comparison for our approach.[9]

Like the other parsers discussed in Rimell et al. (2009) and Nivre et al. (2010), the overall performance on these long-distance constructions is much lower than the overall scores for this parser. However, the parser recovers long-distance dependencies at least as well as standard statistical dependency parsers that use a post-processing step, and better than standard statistical parsers.[10]

[9]Other parsers were evaluated in Rimell et al. (2009), with worse results than what reported here. However, because of differences in set up and parsing architecture, comparing results here would be misleading. For example, the Stanford parser was evaluated, reaching 38% recall. But it should be borne in mind that this result is not directly comparable, as it is likely that this parser too would have benefitted from the post-processing step used in Nivre et al. (2010) to evaluate dependency parsers.

[10]Manual inspection indicates that if we allowed more complex dependencies, such as those proposed by Nivre et al.'s evaluation, our score on subject relative clauses would increase from 57% to 69%, for a total of 49% correct. This explains in part the apparent difference between our architecture and other dependency parsers for subject relative clauses.

The differences in recall between manual and automatic evaluation in Table 1 show that the automatic evaluation is sometimes too strict and sometimes too lenient. The former cases arise primarily in small clause dependencies and dependency recovery by coordination across all LDD constructions, which were taken into account in the manual evaluation, but not in the automatic evaluation, because, as indicated above, scoring coordination automatically is too difficult. This explains the recall difference between the two evaluation methods in SRC and SEmb. The latter case is due to the stricter definition of head in the manual evaluation. This is the main reason why ORed and OQ have lower recall in this evaluation.

Table 2 reports some of the labelled error counts of the most frequent labels. In general, the confusion matrix shows that the labelled correspondence is accurate, and that it corresponds to meaningful generalisations. As can also be observed, a single grammatical function label corresponds to several different semantic relations and vice versa. Full recovery of argument structure, then, requires both grammatical syntactic relations and semantic role labelling.

5.1 Error Analysis of Development Sets

We classify the errors made by our parser on the development set based on Nivre et al. (2010)'s three main error categories, Global, Arg, Link, with some more restrictive modifications that are appropriate for the two-level representation. Following Nivre et al. (2010), we define a Global error as one that applies to cases where the parser fails to build the relevant clausal structure (e.g., the relative clause and what it modifies in ORed, ORC, Free, and SRC) due to parsing/tagging errors. We split Nivre et al.'s definition of Arg errors (errors on labels) in two cases. An Arg error is

	Tot	G	Arg	S	SA	L	Dep
ORed	10	3	4	3	0	0	23
ORC	14	9	2	3	0	0	20
Free	8	3	5	0	0	0	22
OQ	23	14	7	1	1	0	25
RNR	15	1	2	3	0	9	28
SEm	10	4	4	2	0	0	13
SRC	16	12	0	4	0	0	43

Table 3: Distribution of error types in the development sets. (G= Global; S= Sem; SA= Sem+Arg; L= Link; Dep= number of dependencies).

	Us	MST	Malt	Dep
ORed	10	9	13	23
ORC	14	13	16	20
Free	8	5	6	22
OQ	23	17	20	25
RNR	15	14	15	28
SEm	10	9	9	13
SRC	16	10	14	43

Table 4: Comparison of the three dependency parsers based on the total number of errors in each development set.

one which occurs when the parser fails to assign the correct functional relation (e.g., subject, object), while a Sem error is one in which the parser fails to assign the correct semantic relation (e.g., A1, A2). Nivre et al's Link error is one where the parser fails to find a dependency by coordination in the case of right node raising.

Our restrictive modifications follow the constraints indicated above on what counts as a correct dependency. In particular, we only count as correct two types of dependencies: simple, in which the dependency is represented as a single arc in the parse tree; and complex, where a gold dependency corresponds to a path of only two direct dependencies, such as in the case of predicative constructions and prepositional phrases discussed above. Our definition of complex dependencies is stricter than Nivre et al.'s, and we do not count indirect dependencies. Link errors related to relative clauses (indirect dependencies) are classified as Sem errors.[11]

Table 3 shows the frequency of the error types for our parser in the seven development sets. Global errors are most frequent for OQ, ORC and SRC. Questions (OQ) are not well represented in our training data, since they do not include the additional QB data (Nivre et al., 2010) used to improve the performance of MSTParser and MaltParser (see Table 4 for comparison of number of errors for each parser). With respect to ORC and SRC, most Global errors are related to part-of-speech tagging errors and wrong head assignment of complex NPs which are modified by the relevant relative clause. In particular there seems to be a strong recency preference, which assigns the relative clause to the closest noun head in a complex NP. A closer look at Arg errors shows that, in ORed, ORC and OQ, the most frequent errors are because the parser fails to find the Arg relation between a preposition and its argument in cases of preposition stranding.

Based on the comparison of errors of other statistical dependency parsers on the development set, shown in Table 4, we can conclude that the trends of errors by constructions are the same in all three parsers.

6 Conclusions and Future Work

In this paper, we have evaluated an approach to learn two-level long-distance representations that encode argument structure information directly, as a particularly difficult test case, and shown that we can learn these difficult constructions as well as dependency parsers augmented with a dedicated long-distance dependency post-processing step. This work also shows that resources and methods to recover these richer representations already exist.

It is important to recall that the predicate-argument structure of a clause is considered central for NLP applications because it represents the grammatically relevant lexical semantic content of the clause. The two-level parser described in this paper can recover this information, while purely syntactic parsers, whether they recover long-distance dependencies or not, would still need further enhancements.

References

Stephen Clark and James Curran. 2007. Wide-coverage efficient statitstical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

[11] Nivre et al.'s Link errors also include cases where the parser fails to find the crucial Link relations *rcmod* in ORed, ORC, SRC, and SEmb. This type of Link error is not relevant for us.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592, Reykjavik, Iceland, May. ACL Anthology Identifier: L14-1045.

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. 2009. A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 37–42, Boulder, Colorado, June.

Jan Hajic. 1998. Building a syntactically annotated corpus: the Prague dependency treebank. In Eva Hajicova, editor, *Issues of Valency and Meaning*, pages 106–132. Karolinum, Prague.

Eva Hajičová, Jiří Havelka, Petr Sgall, Kateřina Veselá, and Daniel Zeman. 2004. Issues of Projectivity in the Prague Dependency Treebank. *Prague Bulletin of Mathematical Linguistics*, (81).

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 (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):950–998.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143, Philadelphia, Pennsylvania, USA, July.

Beth Levin. 1993. *English Verb Classes and Alternations*. University of Chicago Press, Chicago, IL.

Xavier Lluís and Lluís Màrquez. 2008. A joint model for parsing syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 188–192, Manchester, England, August.

Mitch Marcus, Beatrice Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

Ryan McDonald. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Paola Merlo and Gabriele Musillo. 2008. Semantic parsing for high-precision semantic role labelling. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CONLL-08)*, pages 1–8, Manchester, UK.

Paola Merlo and Lonneke van der Plas. 2009. Abstraction and generalisation in semantic role labels: PropBank, VerbNet or both? In *Procs of ACL and AFNLP*, pages 288–296, Suntec, Singapore, August.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43th Meeting of the ACL*, pages 83–90, Ann Arbor, Michigan.

Luan Nguyen, Marten Van Schijndel, and William Schuler. 2012. Accurate unbounded dependency recovery using generalized categorial grammars. In *Proceedings of COLING 2012*, pages 2125–2140, Mumbai, India, December.

Joakim Nivre, J. Hall, and J. Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, pages 2216–2219.

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gómez Rodríguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 833–841, Beijing, China, August.

Petr Pajas and Jan Štěpánek. 2008. Recent advances in a feature-rich framework for treebank annotation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 673–680, Manchester, UK.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31:71–105.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 813–821, Singapore, August. Association for Computational Linguistics.

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 12th Conference on Computational Natural Language Learning (CoNLL-2008)*, pages 159–177.