Accurate SHRG-Based Semantic Parsing

Yufei Chen, Weiwei Sun and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{yufei.chen,ws,wanxiaojun}@pku.edu.cn

Abstract

We demonstrate that an SHRG-based parser can produce semantic graphs much more accurately than previously shown, by relating synchronous production rules to the syntacto-semantic composition process. Our parser achieves an accuracy of 90.35 for EDS (89.51 for DMRS) in terms of ELEMENTARY DEPENDENCY MATCH, which is a 4.87 (5.45) point improvement over the best existing data-driven model, indicating, in our view, the importance of linguistically-informed derivation for data-driven semantic parsing. This accuracy is equivalent to that of English Resource Grammar guided models, suggesting that (recurrent) neural network models are able to effectively learn deep linguistic knowledge from annotations.

1 Introduction

Graph-structured semantic representations, e.g. Semantic Dependency Graphs (SDG; Clark et al., 2002; Ivanova et al., 2012), Elementary Dependency Structure (EDS; Oepen and Lønning, 2006), Abstract Meaning Representation (AMR; Banarescu et al., 2013), Dependency-based Minimal Recursion Semantics (DMRS; Copestake, 2009), and Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013), provide a lightweight yet effective way to encode rich semantic information of natural language sentences (Kuhlmann and Oepen, 2016). Parsing to semantic graphs has been extensively studied recently.

At the risk of oversimplifying, work in this area can be divided into three types, according to how much structural information of a target graph is explicitly modeled. Parsers of the first type throw an input sentence into a sequence-to-sequence model and leverage the power of deep learning technologies to obtain auxiliary symbols to transform the output sequence into a graph (Peng et al., 2017b; Konstas et al., 2017). The strategy of the second type is to gradually generate a graph in a greedy search fashion (Zhang et al., 2016; Buys and Blunsom, 2017). Usually, a transition system is defined to handle graph construction. The last solution explicitly associates each basic part with a target graph score, and casts parsing as the search for graphs with the highest sum of partial scores (Flanigan et al., 2014; Cao et al., 2017). Although many parsers achieve encouraging results, they are very hard for linguists to interpret and understand, partially because they do not explicitly model the syntacto-semantic composition process which is a significant characteristic of natural languages.

In theory, Synchronous Hyperedge Replacement Grammar (SHRG; Drewes et al., 1997) provides a mathematically sound framework to construct semantic graphs. In practice, however, initial results on the utility of SHRG for semantic parsing were somewhat disappointing (Peng et al., 2015; Peng and Gildea, 2016). In this paper, we show that the performance that can be achieved by an SHRG-based parser is far higher than what has previously been demonstrated. We focus here on relating SHRG rules to the syntacto-semantic composition process because we feel that information about syntax-semantics interface has been underexploited in the data-driven parsing architecture. We demonstrate the feasibility of inducing a high-quality, linguistically-informed SHRG from compositional semantic annotations licensed by English Resource Grammar (ERG; Flickinger, 2000), dubbed English Resource Semantics[1] (ERS). Coupled with RNN-based pars-

[1] http://moin.delph-in.net/ErgSemantics

Model	Grammar	SDG	EDS	DMRS
Data-driven	NO	89.4	85.48	84.16
ERG-based	Unification	92.80	89.58	89.64
SHRG-based	Rewriting	- -	90.39	89.51

Table 1: Parsing accuracy of the best existing grammar-free and -based models as well as our SHRG-based model. Results are copied from (Oepen et al., 2015; Peng et al., 2017a; Buys and Blunsom, 2017).

ing techniques, we build a robust SHRG parser that is able to produce semantic analysis for all sentences. Our parser achieves an accuracy of 90.35 for EDS and 89.51 for DMRS in terms of ELEMENTARY DEPENDENCY MATCH (EDM) which outperforms the best existing grammar-free model (Buys and Blunsom, 2017) by significant margins (see Table 1). This marked result affirms the value of modeling the syntacto-semantic composition process for semantic parsing.

On sentences that can be parsed by ERG-guided parsers, e.g. PET[2] or ACE[3], a significant accuracy gap between ERG-guided parsers and data-driven parsers is repeatedly reported (see Table 1). The main challenge for ERG-guided parsing is limited coverage. Even for treebanking on WSJ sentences from PTB, such a parser lacks analyses for c.a. 11% of sentences (Oepen et al., 2015). Our parser yields equivalent accuracy to ERG-guided parsers and equivalent coverage, full-coverage in fact, to data-driven parsers. We see this investigation as striking a balance between data-driven and grammar-driven parsing. It is not our goal to argue against the use of unification grammar in high-performance deep linguistic processing. Nevertheless, we do take it as a reflection of two points: (1) (recurrent) neural network models are able to effectively learn deep linguistic knowledge from annotations; (2) practical parsing may benefit from transforming a model-theoretic grammar into a generative-enumerative grammar.

The architecture of our parser has potential uses beyond establishing a strong string-to-graph parser. Our grammar extraction algorithm has some freedom to induce different SHRGs following different linguistic hypothesis, and allows some issues in theoretical linguistics to be empirically investigated. In this paper, we examine the

[2] http://pet.opendfki.de/

[3] http://sweaglesw.org/linguistics/ace/

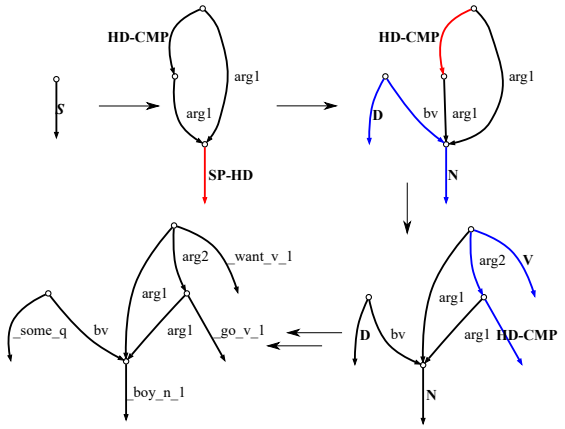


Figure 1: A partial rewriting process of HRG on the semantic graph associated with "Some boys want to go." Lowercase symbols indicate terminal edges, while bold, uppercase symbols indicate nonterminal edges. Red edges are the hyperedges that will be replaced in next step, while the blue edges in next step constitute their corresponding RHS graphs.

lexicalist/constructivist hypothesis, a divide across a variety of theoretical frameworks, in an empirical setup. The lexicalist tradition traces its origins to Chomsky (1970) and is widely accepted by various computational grammar formalisms, including CCG, LFG, HPSG and LTAG. A lexicalist approach argues that the lexical properties of words determine their syntactic and semantic behaviors. The constructivist perspective, e.g. Borer's Exo-Skeletal approach (2005b; 2005a; 2013), emphasizes the role of syntax in constructing meanings. In this paper, we focus on lexicalist and constructivist hypotheses for syntacto-semantic composition. We present our computation-oriented analysis in §6. Under the architecture of our neural parser, a construction grammar works much better than a lexicalized grammar.

2 Hyperedge Replacement Grammar

Hyperedge replacement grammar (HRG) is a context-free rewriting formalism for graph generation (Drewes et al., 1997). An edge-labeled, directed hypergraph is a tuple $H = \langle V, E, l, X \rangle$, where V is a finite set of nodes, and $E \subseteq V^+$ is a finite set of hyperedges. A hyperedge is an extension of a normal edge which can connect to more than two nodes or only one node. $l : E \to L$ assigns a label from a finite set L to each edge. $X \in V^*$ defines an ordered list of nodes, i.e., **ex-**

Algorithm 1 Hyperedge Replacement Grammar Extraction Algorithm

Require: Input syntactic tree T, hypergraph g

RULES $\leftarrow \{\}$

for tree node n in postorder traversal of T **do**

Ensure: Rewriting rule of node n is A $\rightarrow$ B + C, spans of node A, B, C are SPAN-A, SPAN-B, SPAN-C

    SPANS $\leftarrow$ {SPAN-A, SPAN-B, SPAN-C}

    C-EDGES $\leftarrow \{e | e \in \text{EDGES}(g) \wedge \text{SPAN}(e) \in \text{SPANS}\}$

    ALL-NODES $\leftarrow \{s | s \in \text{NODES}(g) \wedge \exists e \in \text{C-EDGES}$ s.t. $s \in \text{NODES}(e)\}$

    S-EDGES $\leftarrow \{e | e \in \text{EDGES}(g) \wedge e$ is structual edge $\wedge \forall s \in \text{NODES}(e) \implies s \in \text{C-EDGES}\}$

    ALL-EDGES = C-EDGES $\cup$ S-EDGES

    INTERNAL-NODES $\leftarrow \{\}$

    EXTERNAL-NODES $\leftarrow \{\}$

    for node s in ALL-NODES **do**

        if $\forall e \in \text{EDGES}(g), s \in \text{NODES}(e) \implies e \in$ ALL-EDGES **then**

            INTERNAL-NODES $\leftarrow$ INTERNAL-NODES $\cup \{s\}$

        else

            EXTERNAL-NODES $\leftarrow$ EXTERNAL-NODES $\cup \{s\}$

        end if

    end for

    RULES $\leftarrow$ RULES $\cup$ {(A, ALL-EDGES, INTERNAL-NODES, EXTERNAL-NODES)}

end for

ternal nodes, which specify the connecting parts when replacing a hyperedge.

An HRG $G = \langle N, T, P, S \rangle$ is a graph rewriting system, where N and T are two disjoint finite sets of nonterminal and terminal symbols respectively. $S \in N$ is the start symbol. P is a finite set of productions of the form $A \rightarrow R$, where the left hand side (LHS) $A \in N$, and the right hand side (RHS) R is a hypergraph with edge labels over $N \cup T$. The rewriting process replaces a nonterminal hyperedge with the graph fragment specified by a production's RHS, attaching each external node to the matched node of the corresponding LHS. An example is shown in figure 1. Following Chiang et al. (2013), we make the nodes only describe connections between edges and store no other information.

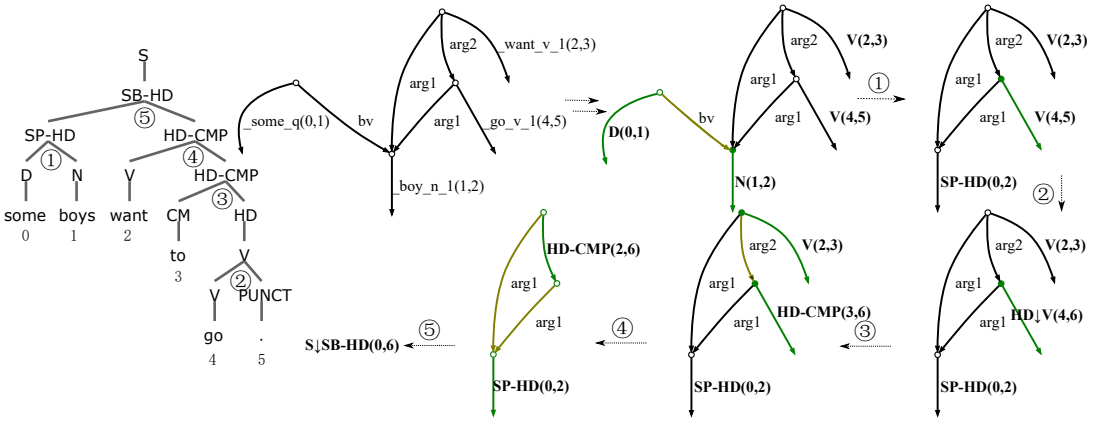
A synchronous grammar defines mappings between different grammars. Here we focus on relating a string grammar, CFG in our case, to a graph grammar, i.e., HRG. SHRG can be represented as tuple $G = \langle N, T, T', P, S \rangle$. N is a finite set of nonterminal symbols in both CFG and HRG. T' and T are finite sets of terminal symbols in CFG and HRG, respectively. $S \in N$ is the start symbol. P is a finite set of productions of the form $A \rightarrow \langle R, R', \sim \rangle$, where $A \in N$, R is a hypergraph fragment with edge labels over $N \cup T$, and R' is a symbol sequence over $N \cup T'$. $\sim$ is a mapping between the nonterminals in R and R'. When a coherent CFG derivation is ready, we can *interpret* it using the corresponding HRG and get a semantic graph.

3 Grammar Extraction

3.1 Graph Representations for ERS

ERS are richly detailed semantic representations produced by the ERG, a hand-crafted, linguistically-motivated HPSG grammar for English. Beyond basic predicate–argument structures, ERS also includes other information about various complex phenomena such as the distinction between scopal and non-scopal arguments, conditionals, comparatives, and many others. ERS are in the formalism of Minimal Recursion Semantics (MRS; Copestake et al., 2005), but can be expressed in different ways. Semantic graphs, including EDS and DMRS, can be reduced from the standard feature structure encoded representations, with or without a loss of information. In this paper, we conduct experiments on ERS data, but our grammar extraction algorithm and the parser are not limited to ERS.

One distinguished characteristic of ERS is that the construction of ERS strictly follows the principle of compositionality (Bender et al., 2015). A precise syntax-semantics interface is introduced



	①	②	③	④	⑤
Shared LHS	SP-HD	HD↓V	HD-CMP	HD-CMP	S↓SP-HD
RHS (syntax)	D + N	V + PUNCT	CM + HD↓V	V + HD-CMP	SP-HD + HD-CMP
RHS (semantics)					

Figure 2: The grammar extraction process of the running example. Conceptual edges which are directly aligned with the syntactic rules are in green. The span-based alignment is shown in arentheses. Structural edges that connect conceptual edges are in brown. Green edges and brown edges together form the subgraph, which acts as RHS in the HRG rule. External nodes are represented as solid dots.

to guarantee compositionality and therefore all meaning units can be traced back to linguistic signals, including both lexical and constructional ones. Take Figure 2 for example. Every concept, e.g. the existence quantifier_some_q, is associated a surface string. We favor such correspondence not because it eases extraction of SHRGs, but because we emphasize sentence meanings that are from forms. The connection between syntax (sentence form) and semantics (word and sentence meaning) is fundamental to the study of language.

3.2 The Algorithm

We introduce a novel SHRG extraction algorithm, which requires and only requires alignments between conceptual edges and surface strings. A tree is also required, but this tree does not have to be a gold-standard syntactic tree. All trees that are compatible with an alignment can be used. The syntactic part of DeepBank is a phrase structure which describes HPSG derivation. The vast majority of syntactic rules in DeepBank are binary, and the rest are unary. In §5, we report evaluation results based on DeepBank trees.

A conceptual graph is composed by two kinds of edges: 1) **conceptual edges** that carry semantic concept information and are connected with only one node, and 2) **structural edges** that build relationships among concepts by connecting nodes. The grammar extraction process repeatedly replaces a subgraph with a nonterminal hyperedge, defining the nonterminal symbol as LHS and the subgraph as RHS. The key problem is to identify an appropriate subgraph in each step. To this end, we take advantage of DeepBank's accurate and fine-grained alignments between the surface string in syntactic tree and concepts in semantic graphs.

To extract the HRG rule synchronized with the syntactic rewriting rule A → B + C, we assume that conceptual edges sharing common spans with A, B or C are in the same subgraph. This subgraph acts as the RHS of the HRG rule. We make the extraction process go in the direction of postorder traversal of the syntactic tree, to ensure that all sub-spans of A, B or C are already replaced with hyperedges. We then add the structural edges that connect the above conceptual edges to RHS. After the subgraph is identified, it is easy to distinguish between internal nodes and external nodes. If all edges connected with a node are in the subgraph, this node is an internal node. Otherwise, it is external node. Finally, the subgraph is replaced with a nonterminal edge connected to all external nodes. Algorithm 1 presents a precise demonstra-

tion and Figure 2 illustrates an example.

4 A Neural SHRG Parser

Under the SHRG formalism, semantic parsing can be divided into two steps: syntactic parsing and semantic interpretation. Syntactic parsing utilizes the CFG part to get a derivation that is shared by the HRG part. At one derivation step, there may be more than one HRG rule applicable. In this case, we need a semantic disambiguation model to choose a good one.

4.1 Syntactic Parsing

Following the LSTM-Minus approach proposed by Cross and Huang (2016), we build a constituent parser with a CKY decoder. We denote the output vectors of forward and backward LSTM as $\boldsymbol{f}_i$ and $\boldsymbol{b}_i$. The feature $\boldsymbol{s}_{i,j}$ of a span (i, j) can be calculated from the differences of LSTM encodings:

$$\boldsymbol{s}_{i,j} = (\boldsymbol{f}_j - \boldsymbol{f}_i) \oplus (\boldsymbol{b}_i - \boldsymbol{b}_j)$$

The operator $\oplus$ means the concatenation of two vectors. Constituency parsing can be regarded as predicting scores for spans and labels, and getting the best syntactic tree with dynamic programming. Follow Stern et al. (2017)'s approach, We calculate the span scores $\textsc{Score}_{\text{span}}(i, j)$ and labels scores $\textsc{Score}_{\text{label}}(i, j, l)$ from $\boldsymbol{s}_{i,j}$ with multilayer perceptrons (MLPs):

$$\textsc{Score}_{\text{span}}(i, j) = \text{MLP}_{\text{span}}(\boldsymbol{s}_{i,j})$$

$$\textsc{Score}_{\text{label}}(i, j, l) = \text{MLP}_{\text{label}}(\boldsymbol{s}_{i,j})[l]$$

$\boldsymbol{x}[i]$ denotes the ith element of a vector $\boldsymbol{x}$. We condense the unary chains into one label to ensure that only one rule corresponds with a specific span. Because the construction rules from DeepBank are either unary or binary, we do not deal with binarization.

Because the SHRG synchronizes at rule level, we need to restrict the parser to ensure that the output agrees with the known rules. The restriction can be directly added into the CKY decoder. To simplify the semantic interpretation process, we add extra label information to enrich the nonterminals in CFG rules. In particular, we consider the count of external nodes of a corresponding HRG rule. For example, the LHS of rule ④ in Figure 2 will be labeled as "`HD-CMP#2`", since the RHS of its HRG counterpart has two external nodes.

4.2 Semantic Interpretation

When a phrase structure tree, i.e., a derivation tree, T is available, semantic interpretation can be regarded as *translating* T to the derivation of graph construction by assigning a corresponding HRG rule to each syntactic counterpart. Our approach to finding the optimal HRG rule combination $\hat{R} = \{r_1, r_2, ...\}$ from the search space $\mathcal{R}(T)$:

$$\hat{R} = \arg\max_{R \in \mathcal{R}(T)} \textsc{Score}(R|T) \quad (1)$$

To solve this optimization problem, we implement a greedy search decoder and a bottom-up beam search decoder. The final semantic graph G is read off from $\hat{R}$.

4.2.1 Greedy Search Model

In this model, we assume that each HRG rule is selected independently of the others. The score of G is defined as the sum of all rule scores:

$$\textsc{Score}(R = \{r_1, r_2, ...\}|T) = \sum_{r \in R} \textsc{Score}(r|T)$$

The maximization of the graph score can be decomposed into the maximization of each rule score. $\textsc{Score}(r|T)$ can be calculated in many ways. Count-based approach is the simplest one, where the rule score is estimated by its frequency in the training data. We also evaluate a sophisticated scoring method, i.e., assigning embedding $\boldsymbol{r}$ for each rule r and training a classifier based on $\boldsymbol{r}$ and $\boldsymbol{s}_{i,j}$:

$$\textsc{Score}(r|T) = \text{MLP}(\boldsymbol{s}_{i,j} \oplus \boldsymbol{r})$$

Inspired by the bag-of-words model, we represent the rule as bag of edge labels. The ith position in $\boldsymbol{r}$ indicates the number of times the i-th label appears in the rule.

4.2.2 Bottom-Up Beam Search Model

We can also leverage structured prediction to approximate $\textsc{Score}(R|T)$ and employ principled decoding algorithms to solve the optimization problem (1). We propose a factorization model to assign scores to the graph and subgraphs in the intermediate state. The score of a certain graph can be seen as the sum of each factor score.

$$\textsc{Score}(R|T) = \sum_{i \in \textsc{Part}(R,T)} \textsc{ScorePart}(i)$$

We use predicates and arguments as factors for scoring. There are two kinds of factors: 1) A conceptual edge aligned with span (i, j) taking predicate name p. We use the span embedding $\boldsymbol{s}_{i,j}$ as features, and scoring with non-linear transformation:

$$\textsc{ScorePart}_{\text{pred}}(i, j, p) = \text{MLP}_{\text{pred}}(\boldsymbol{s}_{i,j})[p]$$

2) A structural edge with label L connects predicates p_a and p_b, which are aligned with spans (i_1, j_1) and (i_2, j_2) respectively. We use the span embeddings $\boldsymbol{s}_{i_1,j_1}$, $\boldsymbol{s}_{i_2,j_2}$ and random initialized predicate embeddings $\boldsymbol{p}_a, \boldsymbol{p}_b$ as features, and scoring with non-linear transformation:

$$\begin{aligned}&\textsc{ScorePart}_{\text{arg}}(i_1, j_1, i_2, j_2, \boldsymbol{p}_a, \boldsymbol{p}_b, L)\\ &\quad = \text{MLP}_{\text{arg}}(\boldsymbol{s}_{i_1,j_1} \oplus \boldsymbol{s}_{i_2,j_2} \oplus \boldsymbol{p}_a \oplus \boldsymbol{p}_b)[L]\end{aligned}$$

We assign a beam to each node in the syntactic tree. To ensure that we always get a subgraph which does not contain any nonterminal edges during the search process, we perform the beam search in the bottom-up direction. We only reserve top k subgraphs in each beam. Figure 3 illustrates the process.

4.3 Training

The objective of training is to make the score of the correct graph higher than incorrect graphs. Our *loss* is calculated from the score difference between the highest-scoring graph and the correct graph, whose corresponding HRG rule combination is R_g:

$$\text{loss} = \max_{R \in \mathcal{R}(T)} \textsc{Score}(R|T) - \textsc{Score}(R_g|T)$$

Following (Kiperwasser and Goldberg, 2016)'s experience of loss augmented inference, in order to update graphs which have high model scores but are very wrong, we augment each factor belonging to the gold graph by adding a penalty term c to its score. Finally the *loss* term becomes:

$$\begin{aligned}\text{loss} = \max_{R \in \mathcal{R}(T)} &(\textsc{Score}(R|T) - \sum_{i \in \textsc{Part}(R,T) \cap \textsc{Part}(R_g,T)} c)\\ &- \textsc{Score}(R_g|T) - \sum_{i \in \textsc{Part}(R_g,T)} c\end{aligned}$$

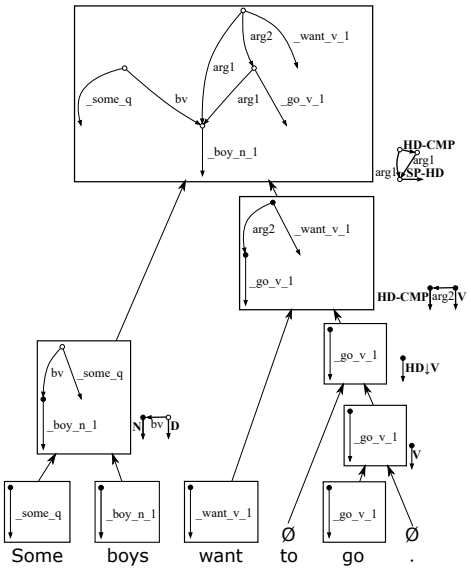


Figure 3: The semantic interpretation process. The interpretation performs bottom-up beam search to get a bunch of high-scored subgraphs for each node in the derivation tree.

5 Experiments

5.1 Set-up

DeepBank is an annotation of the Penn TreeBank Wall Street Journal which is annotated under the formalism of HPSG. We use DeepBank version 1.1, corresponding to ERG 1214, and use the *standard* data split. Therefore the numeric performance can be directly compared to results reported in Buys and Blunsom (2017). We use the pyDelphin library to extract DMRS and EDS graphs and use the jigsaw[4] tool to separate punctuation marks from the words attached to them. We use DyNet[5] to implement our neural models, and automatic batch technique (Neubig et al., 2017) in DyNet to perform mini-batch gradient descent training. The detailed network hyper-parameters can be seen in Table 2. The same pre-trained word embedding as (Kiperwasser and Goldberg, 2016) is employed.

5.2 Results of Grammar Extraction

DeepBank provides fine-grained syntactic trees with rich information. For example, the label `SP-HD_HC_C` indicates that this is a "head+specifier" construction, where the semantic head is also the syntactic head. But there is also the potential for data sparseness. In our ex-

[4] www.coli.uni-saarland.de/~yzhang/files/jigsaw.jar

[5] https://github.com/clab/dynet

Hyperparamter	Value
Batch size	32
Pre-trained word embedding dimension	100
Random-initialized word embedding dimension	150
LSTM Layer count	2
LSTM dimension (each direction)	250
MLP hidden layer count	1
MLP hidden layer dimension	250
penalty term c	1

Table 2: Hyperparamters used in the experiments.

periments, we extract SHRG with three kinds of labels: fine-grained labels, coarse-grained labels and single Xs (i.e. unlabeled parsing). The fine-grained label is the original label, namely fine-grained construction type. We use the part before the first underscore, e.g. SP-HD, as the coarse-grained label. The coarse-grained label is more like the highly generalized rule schemata proposed by Pollard and Sag (1994). Some statistics are shown in Table 3.

Instead of using gold-standard trees to extract synchronous grammar, we also tried randomly-generated alignment-compatible trees. The result shown in Table 4 show that the extracted grammar will be sparse and irregular without correct syntax information.

	#EP	#Rule			#Instance
		Fine	Coarse	Unlabeled	
EDS	1	49689	14234	1476	676817
	2	9616	3424	488	64708
	3	2739	1486	280	11195
	4	1059	732	248	2071
	5+	508	418	251	655
DMRS	1	50668	15745	2688	657999
	2	11428	4418	896	79888
	3	3576	1929	465	14237
	4	1237	873	299	2561
	5+	669	557	297	901

Table 3: Statistics of SHRG rules with different label type by the count of external points in EDS and DMRS representations.

5.3 Results of Syntactic Parsing

In addition to the standard evaluation method for phrase-structure parsing, we find a more suitable measurement, namely condensed score, for our task. Because we condense unary rule chains into one label and extract synchronous grammar under this condensed syntactic tree, it is better to calculate the correctness of a condensed label rather than a single label. The additional label "#N" that indicates the number of external points is also considered in our condensed score evaluation method. The result is shown in Table 5.

Tree Type	1	2	3	4	5+
Gold	1476	488	280	248	251
Fuzzy 1	12710	7591	7963	6578	8998
Fuzzy 2	13606	7355	7228	6090	9112
Fuzzy 3	12278	8228	8462	7039	9946

Table 4: Comparison of grammars extracted from unlabeled gold trees and randomly-generated alignment-compatible trees ("Fuzzy" trees).

Label	Standard				Condensed	
	P	R	F	POS	BCKT	POS
Fine	90.81	91.19	91.00	94.40	87.09	92.98
Coarse	90.78	91.24	91.01	98.30	87.93	95.98

Table 5: Accuracy of syntactic parsing under different labels on development data. We add the count of external nodes of corresponding HRG rule. "POS" concerns the prediction of pre-terminals, while "BCKT" denotes bracketing.

5.4 Results of Semantic Interpretation

We reuse the word embeddings and bidirectional LSTM in the trained syntactic parsing model to extract span embedding $\boldsymbol{s}_{i,j}$. The results of the count-based model, the rule embedding model and the structured model with beam decoder are summarized in Table 6. We report the standard EDM metrics proposed by Dridan and Oepen (2011). The count-based model can achieve considerably good results, showing the correctness of our grammar extraction method. We also try different labels for syntactic trees. The result is shown in Table 7. Models based on the coarse-grained labels achieve optimal performance. The results on the test set of EDS data are shown in Table 8. We achieve state-of-the-art performance with a remarkable improvement over Buys and Blunsom (2017)'s neural parser.

6 On Syntax-Semantics Interface

In this paper, we empirically study the lexicalist/constructivist hypothesis, a divide across a variety of theoretical frameworks, taking semantic parsing as a case study. Although the original grammar that guides the annotation of ERS data,

Model	EDM_P	EDM_A	EDM
Count Based	90.12	81.96	86.03
Rule Embedding	93.41	84.84	89.11
Beam Search	93.48	87.88	90.67

Table 6: The EDM score on EDS development data with different model: count based greedy search, rule embedding greedy search and beam search. We use syntactic trees with coarse-grained labels.

Data	Label	EDM_P	EDM_A	EDM
EDS	Fine	92.70	87.77	90.23
	Coarse	93.48	87.88	90.67
DMRS	Fine	92.52	86.47	89.46
	Coarse	93.60	86.62	90.07

Table 7: Accuracy on the development data under different labels of syntactic tree and beam search.

namely ERG, is highly lexicalized in that the majority of information is encoded in lexical entries (or lexical rules) as opposed to being represented in constructions (i.e., rules operating on phrases), our grammar extraction algorithm has some freedom to induce different SHRGs that choose between lexicalist and constructivist approaches. We modify algorithm 1 to follow the key insights of the lexicalist approach. This is done by considering all outgoing edges when finding the subgraph of the lexical rules. The differences between two kinds of grammars is shown in Table 9.

Different grammars allow the lexicalist/constructivist issue in theoretical linguistics to be empirically examined. The comparison of the counts of rules in each grammar is summarized in Table 11, from which we can see that the sizes of the grammars are comparable. However, the parsing results are quite different, as shown in Table 10. A construction grammar works much better than a lexicalized grammar under the architecture of our neural parser. We take this comparison as informative since lexicalist approaches are more widely accepted by various computational grammar formalisms, including CCG, LFG, HPSG and LTAG.

We think the success of applying SHRG to resolve semantic parsing highly relies on the compositionality nature of ERS' sentence-level semantic annotation. This is the property that makes sure the extracted rules are consistent and regular. Previous investigation by Peng et al. (2015)

	Model	EDM_P	EDM_A	EDM
EDS	Buys and Blunsom	88.14	82.20	85.48
	ACE	91.82	86.92	89.58
	Ours	93.15	87.59	90.35
DMRS	Buys and Blunsom	87.54	80.10	84.16
	ACE	92.08	86.77	89.64
	Ours	93.11	86.01	89.51

Table 8: Accuracy on the test set. We use syntactic trees of coarse-grained labels and beam search.

on SHRG-based semantic parsing utilizes AMRBank which lacks this property to some extent (see Bender et al.'s argument). We think this may be one reason for the disappointing parsing performance. Think about the AMR graph associated with "John wants Bob to believe that he saw him." The AMR annotation for co-reference is a kind of non-compositional speaker meaning, and results in grammar sparseness.

7 On Deep Linguistic Knowledge

Semantic annotations have a tremendous impact on semantic parsing. In parallel with developing new semantic annotations, e.g. AMRBank, there is a resurgence of interest in exploring existing annotations grounded under deep grammar formalisms, such as semantic analysis provided by ERS (Flickinger, 2000). In stark contrast, it seems that only the annotation results gain interests, but not the core annotation engine—knowledge-extensive grammar.

The tendency to continually ignore the positive impact of precision grammar on semantic parsing is somewhat strange. For sentences that can be parsed by an ERG-guided parser, there is a significant accuracy gap which is repeatedly reported. See Table 1 for recent results. The main challenges for precision grammar-guided parsing are unsatisfactory coverage and efficiency that limit their uses in NLP applications. Even for treebanking on newswire data, i.e., Wall Street Journal data from Penn TreeBank (Marcus et al., 1993), ERG lacks analyses for c.a. 11% of sentences (Oepen et al., 2015). For text data from the web, e.g. tweets, this problem is even more serious. Moreover, checking all possible linguistic constraints makes a grammar-guided parser too slow for many realistic NLP applications. Robustness and efficiency, thus, are two major problems for practical NLP applications.

Recent encouraging progress achieved with

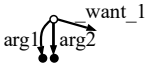
Lexicon	Construction	Lexicalized	CFG Counterpart	Construction	Lexicalized
some			SP-HD → D + N		
want			HD-CMP → V + HD-CMP		
go			S↓SP-HD → SP-HD + HD-CMP		

Table 9: Rules of lexicalized and construction grammars that are extracted from the running example.

Grammar	EDM_P	EDM_A	EDM
Construction	93.48	87.88	90.67
Lexicalized	92.14	81.05	86.63

Table 10: The EDM score on EDS development data with construction grammar and lexicalized grammar using syntax trees of coarse-grained labels and beam search.

Grammar	1	2	3	4	5+
Construction	14234	3424	1486	732	418
Lexicalized	11653	5938	2358	396	11

Table 11: Comparison of the construction grammar and the lexicalized grammar extracted from EDS data. We use syntax trees of coarse-grained labels.

purely data-driven models helps resolve the above two problems. Nevertheless, it seems too radical to remove all explicit linguistic knowledge about the syntacto-semantic composition process, the key characteristics of natural languages. In this paper, we introduce a neural SHRG-based semantic parser that strikes a balance between data-driven and grammar-guided parsing. We encode deep linguistic knowledge partially in a symbolic way and partially in a statistical way. It is worth noting that the symbolic system is a derivational, generative-enumerative grammar, while the origin of the data source is grounded under a representational, model-theoretic grammar. While grammar writers may favor the convenience provided by a unification grammar formalism, a practical parser may re-use algorithms by another formalism by *translating* grammars through *data*. Experiments also suggest that (recurrent) neural network models are able to effectively gain some deep linguistic knowledge from annotations.

8 Conclusion

The advantages of using graph grammars to resolve semantic parsing is clear in concept. However, previous work fails to take these advantages to build a high-performance parser. We have shown ways to improve SHRG-based string-to-semantic-graph parsing.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61772036, 61331011) and the Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for their helpful comments. Weiwei Sun is the corresponding author.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 228–238. http://www.aclweb.org/anthology/P13-1023.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, pages 178–186. http://www.aclweb.org/anthology/W13-2322.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann A. Copestake. 2015. Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th International Conference on Computational Semantics, IWCS 2015, 15-17 April, 2015, Queen Mary*

University of London, London, UK. pages 239–249. http://aclweb.org/anthology/W/W15/W15-0128.pdf.

H. Borer. 2005a. *In Name Only*. Hagit Borer. Oxford University Press. https://books.google.com/books?id=cAEmAQAAIAAJ.

H. Borer. 2005b. *The Normal Course of Events*. Hagit Borer. Oxford University Press. https://books.google.com/books?id=M48UPLst_MQC.

H. Borer. 2013. *Structuring Sense: Volume III: Taking Form*. Borer, Hagit. OUP Oxford. https://books.google.com/books?id=tUkGAQAAQBAJ.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1215–1226. http://aclweb.org/anthology/P17-1112.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017. Parsing to 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 2110–2120. http://aclweb.org/anthology/P17-1193.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 924–932. http://www.aclweb.org/anthology/P13-1091.

Noam Chomsky. 1970. Remarks on nominalization. In R. A. Jacobs and P. S. Rosenbaum, editors, *Readings in English Transformational Grammar*, Waltham, MA, pages 170–221.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures using a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*. pages 327–334. http://www.aclweb.org/anthology/P02-1042.pdf.

Ann Copestake. 2009. ***Invited Talk:*** slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*. Association for Computational Linguistics, Athens, Greece, pages 1–9. http://www.aclweb.org/anthology/E09-1001.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal Recursion Semantics: An introduction. *Research on Language and Computation* pages 281–332.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1–11. https://aclweb.org/anthology/D16-1001.

F. Drewes, H.-J. Kreowski, and A. Habel. 1997. Handbook of graph grammars and computing by graph transformation. World Scientific Publishing Co., Inc., River Edge, NJ, USA, chapter Hyperedge Replacement Graph Grammars, pages 95–162. http://dl.acm.org/citation.cfm?id=278918.278927.

Rebecca Dridan and Stephan Oepen. 2011. Parser evaluation using elementary dependency matching. In *Proceedings of the 12th International Conference on Parsing Technologies*. Association for Computational Linguistics, Dublin, Ireland, pages 225–230. http://www.aclweb.org/anthology/W11-2927.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1426–1436. http://www.aclweb.org/anthology/P14-1134.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Nat. Lang. Eng.* 6(1):15–28.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*. Jeju, Republic of Korea, pages 2–11.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 146–157. http://aclweb.org/anthology/P17-1014.

Marco Kuhlmann and Stephan Oepen. 2016. Towards a catalogue of linguistic graph banks. *Computational Linguistics* 42(4):819–827.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics* 19(2):313–330. http://dl.acm.org/citation.cfm?id=972470.972475.

Graham Neubig, Yoav Goldberg, and Chris Dyer. 2017. On-the-fly operation batching in dynamic computation graphs. In *Advances in Neural Information Processing Systems*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresová. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based mrs banking. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*. European Language Resources Association (ELRA), Genoa, Italy. ACL Anthology Identifier: L06-1214.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017a. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 2037–2048. http://aclweb.org/anthology/P17-1186.

Xiaochang Peng and Daniel Gildea. 2016. Uofr at semeval-2016 task 8: Learning synchronous hyperedge replacement grammar for amr parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 1185–1189. http://www.aclweb.org/anthology/S16-1183.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for amr parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Beijing, China, pages 32–41. http://www.aclweb.org/anthology/K15-1004.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017b. Addressing the data sparsity issue in neural amr parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 366–375. http://www.aclweb.org/anthology/E17-1035.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 818–827. http://aclweb.org/anthology/P17-1076.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics* 42(3):353–389. http://aclweb.org/anthology/J16-3001.