Discourse Representation Parsing for Sentences and Documents

Jiangming Liu **Shay B. Cohen** **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
Jiangming.Liu@ed.ac.uk, {scohen,mlap}@inf.ed.ac.uk

Abstract

We introduce a novel semantic parsing task based on Discourse Representation Theory (DRT; Kamp and Reyle 1993). Our model operates over Discourse Representation Tree Structures which we formally define for sentences and documents. We present a general framework for parsing discourse structures of arbitrary length and granularity. We achieve this with a neural model equipped with a supervised hierarchical attention mechanism and a linguistically-motivated copy strategy. Experimental results on sentence- and document-level benchmarks show that our model outperforms competitive baselines by a wide margin.

1 Introduction

Semantic parsing is the task of mapping natural language to machine interpretable meaning representations. Various models have been proposed over the years to learn semantic parsers from linguistic expressions paired with logical forms, SQL queries, or source code (Kate et al., 2005; Liang et al., 2011; Zettlemoyer and Collins, 2005; Banarescu et al., 2013; Wong and Mooney, 2007; Kwiatkowski et al., 2011; Zhao and Huang, 2015).

The successful application of encoder-decoder models (Sutskever et al., 2014; Bahdanau et al., 2015) to a variety of NLP tasks has prompted the reformulation of semantic parsing as a sequence-to-sequence learning problem (Dong and Lapata, 2016; Jia and Liang, 2016; Kočiskỳ et al., 2016), although most recent efforts focus on architectures which make use of the syntax of meaning representations, e.g., by developing tree or graph-structured decoders (Dong and Lapata, 2016; Cheng et al., 2017; Yin and Neubig, 2017; Alvarez-Melis and Jaakkola, 2017; Rabinovich et al., 2017; Buys and Blunsom, 2017).

In this work we focus on parsing formal meaning representations in the style of Discourse Representation Theory (DRT; Kamp and Reyle 1993).

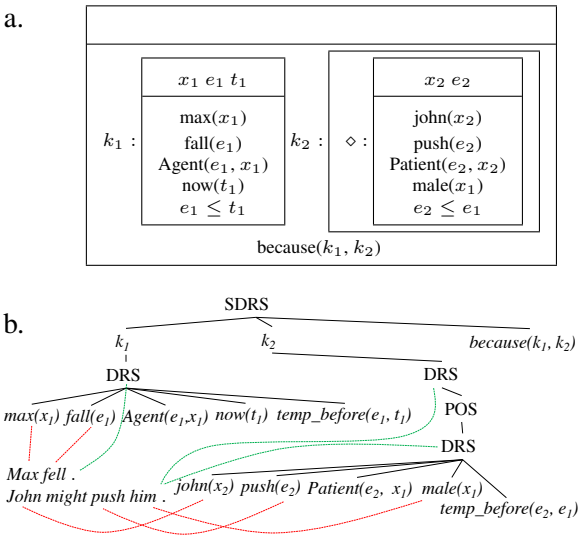

Figure 1: Meaning representation for the discourse "*Max fell. John might push him.*" in box-like format (top) and as a tree (bottom). Red lines indicate terminals corresponding to words and green lines indicate non-terminals corresponding to sentences. $\diamond$ and POS are modality operators for possibility.

DRT is a popular theory of meaning representation (Kamp, 1981; Kamp and Reyle, 1993; Asher, 1993; Asher and Lascarides, 2003) designed to account for a variety of linguistic phenomena, including the interpretation of pronouns and temporal expressions within and across sentences. The basic meaning-carrying units in DRT are Discourse Representation Structures (DRSs) which consist of discourse referents (e.g., x_1, x_2) representing entities in the discourse and discourse conditions (e.g., max(x_1), male(x_1)) representing information about discourse referents. An example of a two-sentence discourse in box-like format is shown in Figure 1a. DRT parsing resembles the task of mapping sentences to Abstract Meaning Representations (AMRs; Banarescu et al. 2013) in that logical forms are broad-coverage, they represent compositional utterances with varied vocabu-

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6248–6262
Florence, Italy, July 28 - August 2, 2019.

lary and syntax and are ungrounded, i.e., they are not tied to a specific database from which answers to queries might be retrieved (Zelle and Mooney, 1996; Cheng et al., 2017; Dahl et al., 1994).

Our work departs from previous general-purpose semantic parsers (Flanigan et al., 2016; Foland and Martin, 2017; Lyu and Titov, 2018; Liu et al., 2018; van Noord et al., 2018b) in that we focus on building representations for *entire* documents rather than *isolated* utterances, and introduce a novel semantic parsing task based on DRT. Specifically, our model operates over Discourse Representation Tree Structures (DRTSs) which are DRSs rendered in a tree-style format (Liu et al. 2018; see Figure 1b). Discourse representation parsing has been gaining more attention lately.[1] The semantic analysis of text beyond isolated sentences can enhance various NLP applications such as information retrieval (Zou et al., 2014), summarization (Goyal and Eisenstein, 2016), conversational agents (Vinyals and Le, 2015), machine translation (Sim Smith, 2017; Bawden et al., 2018), and question anwsering (Rajpurkar et al., 2018).

Our contributions in this work can be summarized as follows: 1) We formally define Discourse Representation Tree structures for sentences and documents; 2) We present a general framework for parsing discourse structures of arbitrary length and granularity; our framework is based on a neural model which decomposes the generation of meaning representations into three stages following a coarse-to-fine approach (Liu et al., 2018; Dong and Lapata, 2018); 3) We further demonstrate that three modeling innovations are key to tree structure prediction: a supervised hierarchical attention mechanism, a linguistically-motivated copy strategy, and constraint-based inference to ensure well-formed DRTS output; 4) Experimental results on sentence- and document-level benchmarks show that our model outperforms competitive baselines by a wide margin. We release our code and DRTS benchmarks in the hope of driving research in semantic parsing further.[2]

[1]The shared task on Discourse Representation Structure parsing in IWCS 2019. https://sites.google.com/view/iwcs2019/home

[2]https://github.com/LeonCrashCode/TreeDRSparsing

2 Discourse Representation Trees

In this section, we define Discourse Representation Tree Structures (DRTSs). We adopt the box-to-tree conversion algorithm of Liu et al. (2018) to obtain trees which we generalize to multi-sentence discourse. As shown in Figure 1, the conversion preserves most of the content of DRS boxes, such as referents, conditions, and their dependencies. Furthermore, we add alignments between sentences and DRTSs nodes.

A DRTS is represented by a labeled tree over a domain $\mathbb{D} = [R, V, C, N]$ where R denotes relation symbols, V denotes variable symbols, C denotes constants and N denotes scoping symbols. Variables V are indexed and can refer to entities x, events e, states s, time t, propositions p, and segments k.[3] R is the disjoint union of a set of elementary relations R_e and segment relations R_s. The set N is defined as the union of binary scoping symbols N_b and unary scoping symbols N_u, where $N_b = \{\text{IMP}, \text{OR}, \text{DUP}\}$, denoting conditions involving implication, disjunction, and duplex,[4] and $N_u = \{\text{POS}, \text{NEC}, \text{NOT}\}$ denoting modality operators expressing possibility, necessity, and negation.

There are six types of nodes in a DRTS: *simple* scoped nodes, *proposition* scoped nodes, *segment* scoped nodes, elementary DRS nodes, segmented DRS nodes, and atomic nodes. Atomic nodes are leaf nodes such that their label is an instantiated relation $r \in R$ with argument variables from V or constants from C.[5] Relations can either be unary or binary. For example, in Figure 1, $\text{male}(x_1)$ denotes an atomic node with a unary relation, while $\text{Patient}(e_2, x_1)$ denotes a binary relation node.

A simple scoped node can take one of the labels in N. A node that takes a label from N_u has only one child which is either an elementary or a segmented DRS node. A binary scope label node can take two children nodes which are an elementary or a segmented DRS. A proposition scoped node can take as label one of the proposition variables p. Its children are elementary or segmented DRS nodes. A segment scoped node can take as label one of the segment variables k and its chil-

[3]Segment variables originate from Segmented Discourse Representation Theory (SDRT; Asher and Lascarides 2003), and denote units connected by discourse relations.

[4]Duplex represents wh-questions (e.g., *who, what, how*).

[5]In our formulation, the only constants used are for denoting numbers. Proper names are denoted by relations, such as $\text{John}(x_2)$.

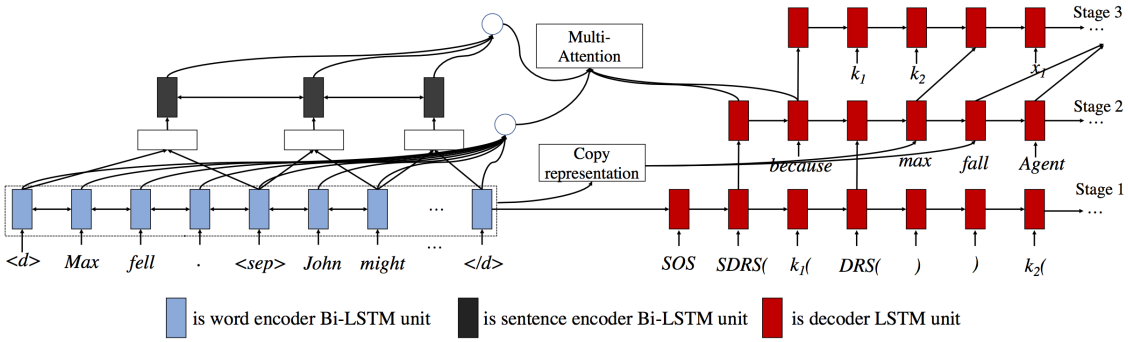

Figure 2: The DRTS parsing framework; words and sentences are encoded with bi-LSTMs; documents are decoded in three stages, starting with tree non-terminals, then relations, and finally variables. Decoding makes use of multi-attention and copying.

dren are elementary or segmented DRS nodes.

An elementary DRS node is labeled with "DRS" and has children (one or more) which are atomic nodes (taking relations from R_e), simple scoped nodes, or proposition scoped nodes. Atomic nodes may use any of the variables except for segment variables k. Finally, a segmented DRS node (labeled with "SDRS") takes at least two children nodes which are segment scoped nodes and at least one atomic node (where the variables allowed are the segment variables that were chosen for the other children nodes and the relations are taken from R_s). For example, the root node in Figure 1 is an SDRS node with two segment variables k_1 and k_2 and the instantiated relation is $\mathrm{because}(k_1, k_2)$. The children of the nodes labeled with the segment variables are elementary or segmented DRS nodes. A full DRTS is a tree with an elementary or segmented DRS node as root.

3 Modeling Framework

We propose a unified framework for sentence- and document-level semantic parsing based on the encoder-decoder architecture shown in Figure 2. The encoder is used to obtain word and sentence representations while the decoder generates trees in three stages. Initially, elementary DRS nodes, segmented DRS nodes, and scoped nodes are generated. Next, the relations of atomic nodes are predicted, followed by their variables. In order to make the framework compatible for discourse structures of arbitrary length and granularity and capable of adopting document-level information, we equip the decoder with multi-attention, a supervised attention mechanism for aligning DRTS nodes to sentences, and a linguistically-motivated copy strategy.

3.1 Encoder

Documents (or sentences) are represented as a sequence of words $\langle d\rangle$,w_{00},...,$\langle sep_i\rangle$,...,w_{ij},...,$\langle /d\rangle$, where $\langle d\rangle$ and $\langle /d\rangle$ denote the start and end of document, respectively, and $\langle sep_i\rangle$ denotes the right boundary of the ith sentence.[6]

The jth token in the ith sentence of a document is represented by vector $x_{ij} = f([e_{w_{ij}}; \bar{e}_{w_{ij}}; e_{\ell_{ij}}])$ which is the concatenation (;) of randomly initialized embeddings $e_{w_{ij}}$, pre-trained word embeddings $\bar{e}_{w_{ij}}$, and lemma embeddings $e_{\ell_{ij}}$ (where $f(\cdot)$ is a non-linear function). Embeddings $e_{w_{ij}}$ and $e_{\ell_{ij}}$ are randomly initialized and tuned during training, while $\bar{e}_{w_{ij}}$ are fixed.

The encoder represents words and sentences in a unified framework compatible with sentence- and document-level DRTS parsing. Our experiments employed recurrent neural networks with long-short term memory units (LSTMs; Hochreiter and Schmidhuber 1997), however, there is nothing inherent in our framework that is LSTM specific. For instance, representations based on convolutional (Kim, 2014) or recursive neural networks (Socher et al., 2012) are also possible.

Word Representation We encode the input text with a bidirectional LSTM (biLSTM):

$$[\overleftrightarrow{h}_{x_{00}} : \overleftrightarrow{h}_{x_{mn}}] = \mathrm{biLSTM}(x_{00} : x_{mn}),$$

where $\overleftrightarrow{h}_{x_{ij}}$ denotes the hidden representation of the encoder for x_{ij}, which denotes the input representation of token j in sentence i.

[6] The left boundary of sentence i is the right boundary of sentence $i-1$, the left boundary of the first sentence is $\langle d\rangle$, and the right boundary of the last sentence is $\langle /d\rangle$.

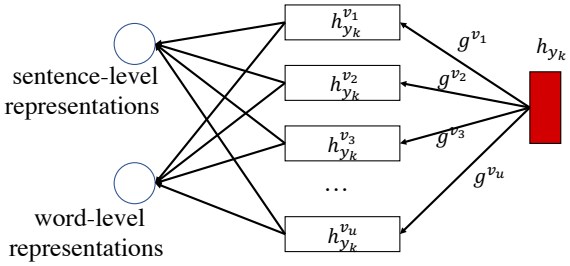

Figure 3: Multi-attention component; linear functions $g^v(\cdot)$ transform decoder hidden representations into different vector spaces, where v shows which linear function is applied, e.g. $h_{y_k}^{\text{word}} = g^{\text{word}}(h_{y_k})$.

Shallow Sentence Representation Each sentence can be represented via the concatenation of the forward hidden state of its right boundary and the backward hidden state of its left boundary, i.e., $h_{x_i} = [\overrightarrow{h_{x_{\langle sep_i \rangle}}}; \overleftarrow{h_{x_{\langle sep_{i-1} \rangle}}}]$.

Deep Sentence Representation An alternative to the shallow sentence representation just described, is a biLSTM encoder:

$$[\overleftrightarrow{h_{x_0}} : \overleftrightarrow{h_{x_m}}] = \text{biLSTM}(h_{x_0} : h_{x_m}),$$

which takes h_{x_i}, the shallow sentence representation, as input.

3.2 Decoder

We generate DRTSs following a three-stage decoding process (Liu et al., 2018), where each stage can be regarded as a sequential prediction on its own. Based on this, we propose the multi-attention mechanism to make it possible to deal with multiple sentences. The backbone of our tree-generation procedure is an LSTM decoder which takes encoder representations H_x as input and constructs bracketed trees (i.e., strings) in a top-down manner, while being equipped with multi-attention. We first describe this attention mechanism as it underlies all generation stages and then move on to present each stage in detail.

3.2.1 Multi-Attention

Multi-attention aims to extract features from *different* encoder representations and is illustrated in Figure 3. The hidden representations h_{y_k} of the decoder are fed to various linear functions to obtain vector space representations:

$$h_{y_k}^v = g^v(h_{y_k}),$$

where $g^v(\cdot)$ is a linear function with the name v.[7] Given encoder representations $H_x = h_{x_0}, h_{x_1}, ...h_{x_m}$, we extract features by applying a standard attention mechanism (Bahdanau et al., 2015) on encoder representations $h_{y_k}^v$:

$$\text{Attn}^v(h_{y_k}, H_x) = \text{Attn}(h_{y_k}^v, H_x) = \sum_{i=1}^{m} \beta_{ki}^v h_{x_i},$$

where weight β_{ki}^v is computed by:

$$\beta_{ki}^v = \frac{\exp(h_{y_k}^{v\text{T}} h_{x_i})}{\sum_o \exp(h_{y_k}^{v\text{T}} h_{x_o})}.$$

Multi-attention scores can be also obtained from the attention weights:

$$\text{Score}^v(h_{y_k}, H_x) = [\beta_{k0}^v : \beta_{km}^v]$$

3.2.2 Tree Generation

Stage 1 Our decoder first generates tree nonterminals $y_0^{\text{st}}, ..., y_k^{\text{st}}$ (see Figure 2).[8] The probabilistic distribution of the kth prediction is:

$$P(y_k^{\text{st}} | y_{<k}^{\text{st}}, H_x) = \text{SoftMax}(s_k^{\text{st}}),$$

where H_x refers to the encoder representations and score s_k^{st} is computed as:

$$\begin{aligned} s_k^{\text{st}} = f([h_{y_k^{\text{st}}}; \text{Attn}^{\text{word}}(h_{y_k^{\text{st}}}, [h_{x_{00}} : h_{x_{mn}}]) \\ ; \text{Attn}^{\text{sent}}(h_{y_k^{\text{st}}}, [h_{x_0} : h_{x_m}])]), \end{aligned} \quad (1)$$

where $h_{y_k^{\text{st}}}$ is the hidden representation of the decoder in Stage 1, i.e., $h_{y_k^{\text{st}}} = \text{LSTM}(e_{y_{k-1}^{\text{st}}})$.[9]

Stage 2 Given elementary or segmented DRS nodes generated in Stage 1, atomic nodes $y_0^{\text{nd}}, ..., y_k^{\text{nd}}$ are predicted (see Figure 2), with the aid of copy strategies which we discuss shortly. The probabilistic distribution of the kth prediction is:

$$P(y_k^{\text{nd}} | y_{<k}^{\text{nd}}, H_x, H_{y^{\text{st}}}) = \text{SoftMax}([s_k^{\text{nd}}; s_k^{\text{copy}}]),$$

where s_k^{nd} and s_k^{copy} are generation and copy scores, respectively, over the kth prediction.

$$\begin{aligned} s_k^{\text{nd}} = f([h_{y_k^{\text{nd}}}; \text{Attn}^{\text{word}}(h_{y_k^{\text{nd}}}, [h_{x_{00}} : h_{x_{mn}}]) \\ ; \text{Attn}^{\text{sent}}(h_{y_k^{\text{nd}}}, [h_{x_0} : h_{x_m}])]) \end{aligned} \quad (2)$$

[7] In this paper, v could be "word", "sent", "copy", "st2nd" (from first to second stage) and "nd2rd" (from second to third stage), which are used to distinguish linear functions in different roles, as explained later.

[8] Upper subscripts "st", "nd", and "rd" denote Stage 1, 2, and 3, respectively.

[9] y_{-1}^{st} is special token SOS denoting the start of sequence.

$$s_k^{\text{copy}} = \text{Score}^{\text{copy}}(h_{y_k^{\text{nd}}}, [h_{\ell'_0}^{\text{copy}} : h_{\ell'_z}^{\text{copy}}]) \quad (3)$$

where $[h_{\ell_0}^{\text{copy}} : h_{\ell_z}^{\text{copy}}]$ are copy representations used for copy scoring; and $h_{y_k^{\text{nd}}}$ is the hidden representation of the decoder in Stage 2, which is obtained based on how the previous token was constructed:

$$h_{y_k}^{\text{nd}} = \begin{cases} \text{LSTM}(g^{\text{copy}}(h_{y_{k-1}^{\text{nd}}}^{\text{copy}})) & y_{k-1}^{\text{nd}} \text{ is copied} \\ \text{LSTM}(e_{y_{k-1}^{\text{nd}}}) & y_{k-1}^{\text{nd}} \text{ is generated} \\ \text{LSTM}(g^{\text{st2nd}}(h_{\text{drs}})) & k = 0 \end{cases}$$

The generation of atomic nodes in the second stage is conditioned on h_{drs}, the decoder hidden representation of elementary or segmented DRS nodes from Stage 1 by the linear function g^{st2nd}.

For the generation of atomic nodes, we copy lemmas from the input text. However, copying is limited to unary nodes which mostly represent entities and predicates (e.g., john(x_1), eat(e_1)), and correspond almost verbatim to input tokens. Binary atomic nodes denote semantic relations between two variables and do not directly correspond to the surface text. For example, given the DRTS for the utterance "*the oil company is deprived of ...*", nodes oil(x_1) and company(x_2) will be copied from *oil* and *company*, while node of(x_2, x_1) will not be copied from *deprived of*.

Copy representations $M_d = [h_{\ell'_0}^{\text{copy}} : h_{\ell'_z}^{\text{copy}}]$ are constructed for each document d from its encoder hidden representations $[h_{x_{00}} : h_{x_{mn}}]$, by averaging the encoder word representations which have the same lemma, where $\ell' \in L'$ and L' is the set of distinct lemmas in document d:

$$h_{\ell'_z} = \frac{1}{N} \sum_{(ij):\ell_{ij}=\ell'_z} h_{x_{ij}},$$

and N is the number of tokens with lemma ℓ'_z.

Stage 3 Finally, we generate terminals, i.e. atomic node variables $y_0^{\text{rd}}, ..., y_k^{\text{rd}}$ (see Figure 2). The probabilistic distribution of the kth prediction is:

$$P(y_k^{\text{rd}} | y_{<k}^{\text{rd}}, H_x, H_{y^{\text{nd}}}) = \text{SoftMax}(s_k^{\text{rd}}),$$

$$s_k^{\text{rd}} = f([h_{y_k^{\text{rd}}}; \text{Attn}^{\text{word}}(h_{y_k^{\text{rd}}}, [h_{x_{00}} : h_{x_{mn}}]); \text{Attn}^{\text{sent}}(h_{y_k^{\text{rd}}}, [h_{x_0} : h_{x_m}])]) \quad (4)$$

where $h_{y_k^{\text{rd}}}$ is the decoder hidden representation in the third stage:

$$h_{y_k^{\text{rd}}} = \begin{cases} \text{LSTM}(e_{y_{k-1}^{\text{rd}}}) & k \neq 0 \\ \text{LSTM}(g^{\text{nd2rd}}(h_{\text{atm}})) & k = 0 \end{cases}$$

Here, the generation of variables is conditioned upon h_{atm}, the decoder hidden representation of atomic nodes from the second stage by the linear function g^{nd2rd}.

3.3 Training

The model is trained to minimize an average cross-entropy loss objective:

$$L(\theta) = -\frac{1}{N} \sum_j \log p_j, \quad (5)$$

where p_j is the distribution of output tokens, θ are the parameters of the model. We use stochastic gradient descent and adjust the learning rate with Adam (Kingma and Ba, 2014).

4 Extensions

In this section we present two important extensions to the basic modeling framework outlined above. These include a supervised attention mechanism dedicated to aligning sentences to tree nodes. This type of alignment is important when parsing documents (rather than individual sentences) and may also enhance the quality of the copy mechanism. Our second extension concerns the generation of well-formed and meaningful logical forms which is generally challenging for semantic parsers based on sequence-to-sequence architectures, even more so when dealing with long and complex sequences pertaining to documents.

4.1 Supervised Attention

The attention mechanism from Section 3.2.1 can automatically learn alignments between encoder and decoder hidden representations. However, as shown in Figure 1, DRTSs are constructed recursively and alignment information between DRTS nodes and sentences is available. For this reason, we propose a method to explicitly learn this alignment by exploiting the feature representations afforded by multi-attention. Specifically, we obtain alignment weights via multi-attention:

$$\text{Score}^{\text{align}}(h_{y_k}, [h_{x_0} : h_{x_m}]) = [\beta_{k0}^{\text{align}} : \beta_{km}^{\text{align}}]$$

where $\beta_{km}^{\text{align}} = P(a_k = m|h_{y_k}, [h_{x_0} : h_{x_m}])$, i.e., the probabilistic distribution over alignments from sentences to the kth prediction in the decoder, where $a_k = m$ denotes the kth prediction aligned to the mth sentence. We add an alignment loss to the objective in Equation (5):

$$L(\theta) = -\frac{1}{N}\sum_j \log p_j + \frac{1}{N^{\text{align}}}\sum_k \log p_k^{\text{align}},$$

where p_k^{align} is the probability distribution of alignments. We then use these alignments in two ways.

Alignments as Features Alignments are incorporated as additional features in the decoder by concatenating the aligned sentence representations with the scoring layers. Equations (1), (2), and (4) are thus rewritten as:

$$s_k^{stg} = f([h_{y_k^{stg}}; \text{Attn}^{\text{word}}(h_{y_k^{stg}}, [h_{x_{00}} : h_{x_{mn}}]) \; h_{x_{a_k}}; \text{Attn}^{\text{sent}}(h_{y_k^{stg}}, [h_{x_0} : h_{x_m}])]),$$

where $stg \in \{\text{st}, \text{nd}, \text{rd}\}$, and $h_{x_{a_k}}$ is the a_kth sentence representation.

At test time, the scoring layer requires the alignment information, so we first select the sentence with the highest probability, i.e., $a_k^* = \arg\max_{a_k} P(a_k|h_{y_k}, [h_{x_0} : h_{x_m}])$, and then add its representation $h_{x_{a_k}^*}$ to the scoring layer.

Copying from Alignments We use alignment as a means to modulate which information is copied. Specifically, we allow copying to take place only over sentences aligned to elementary DRS nodes. We construct copy representations for each sentence in a document, i.e., $M_0, ..., M_i, ..., M_m$ where $M_i = [h_{\ell'_{i0}}^{\text{copy}} : h_{\ell'_{iz}}^{\text{copy}}]$, $\ell'_{iz} \in L'_i$, and L'_i is the set of distinct lemmas in the ith sentence:

$$h_{\ell'_{iz}}^{\text{copy}} = \frac{1}{N}\sum_{(ij):\ell_{ij}=\ell'_{iz}} h_{x_{ij}},$$

Given the alignment between elementary DRS nodes and sentences, we calculate the copying score by rewriting Equation (3) as:

$$s_k^{\text{copy}} = \text{Score}^{\text{copy}}(h_{y_k^{\text{nd}}}, M_a)$$

where a is the index of the sentence that is aligned to the elementary DRS node.

At test time, when an elementary DRS is generated during the first stage, we further predict which sentence the node should be aligned to. The information is then passed onto the second stage, and elements from the aligned sentence can be copied.

step	stack	valid candidates	prediction
1	[]	SDRS(, DRS(	SDRS(
2	[SDRS($_0$]	k_1(	k_1(
3	[SDRS($_0$, k_1($_0$]	SDRS(, DRS(	DRS(
4	[SDRS($_0$, k_1($_0$, DRS(]	*simp*SNs,)	)
5	[SDRS($_0$, k_1($_1$]	)	)
6	[SDRS($_1$]	k_2(	k_2(
...	...	...	...

Figure 4: Constraint-based inference in decoding stage 1; *simp*SNs are simple scoped nodes; subscripts denote the number of children already constructed.

4.2 Constraint-based Inference

Recall that our decoder consists of three stages, each of which is a sequence-to-sequence model. As a result, there is no guarantee that tree output will be well-formed. To ensure the generation of syntactically valid trees, at each step, we generate the set of valid candidates Y_k^{valid} which do not violate the DRTS definitions in Section 2, and then select the highest scoring tree as our prediction:

$$y_k^* = \arg\max_{y_k \in Y_k^{\text{valid}}} P(y_k|y_{<k}, \theta),$$

where θ are the parameters of the model, and Y_k^{valid} the set of valid candidates at step k.

In Stage 1, partial DRTSs are stored in a stack and for each prediction the model checks the stack to obtain a set of valid candidates. In the example in Figure 4, segment scoped node k_1 has a child already at step 5, so predicting a right bracket would not violate the definition of DRTS.[10] In stage 2, when generating relations for elementary DRS nodes, the candidates come from R_e and lemmas that are used for copying; when generating relations for segmented DRS nodes, the candidates only come from R_s. Finally, in stage 3 we generate only two variables for binary relations and one variable for unary relations. A formal description is given in the Appendix.

5 Experimental Setup

Benchmarks Our experiments were carried out on the Groningen Meaning Bank (GMB; Bos et al. 2017) which provides a large collection of English texts annotated with Discourse Representation Structures. We preprocessed the GMB into the tree-based format defined in Section 2 and created two benchmarks, one which preserves document-level boundaries, and a second one which treats sentences as isolated instances. Various statistics

[10] Similar constraints apply to unary simple scoped nodes.

	Sentences		Documents			
	#sent	avg_w	#doc	#sent	avg_s	avg_w
train	41,563	21.1	7,843	48,599	6.2	135.3
dev	5,173	21.0	991	6,111	6.2	134.0
test	5,451	21.2	1,035	6,469	6.3	137.2

Table 1: Statistics on the GMB sentence- and document-level benchmarks (avg_w denotes the average number of words per sentence (or document), avg_s denotes the average number of sentences per document).

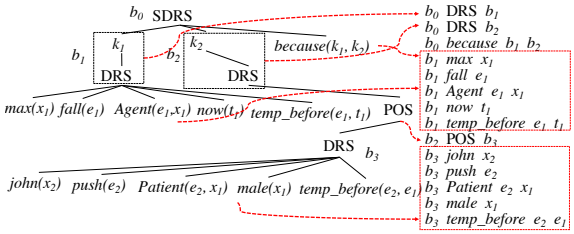

Figure 5: Clausal form for DRTS corresponding to the document "*Max fell. John might push him.*".

on these are shown in Table 1, for the respective training, development, and testing partitions. We followed the same data splits as Liu et al. (2018).

Settings We carried out experiments on the sentence- and document-level GMB benchmarks in order to evaluate our framework. We used the same empirical hyper-parameters for sentence- and document-level parsing. The dimensions of word and lemma embeddings were 300 and 100, respectively. The encoder and decoder had two layers with 300 and 600 hidden dimensions, respectively. The dropout rate was 0.1. Pre-trained word embeddings (100 dimensions) were generated with Word2Vec trained on the AFP portion of the English Gigaword corpus.[11]

Model Comparison For the sentence-level experiments, we compared our DRTS parser against Liu et al. (2018) who also perform tree parsing and have a decoder which first predicts the structure of the DRS, then its conditions, and finally its referents. Our parser without the document-level component is similar to Liu et al. (2018); a key difference is that our model is equipped with linguistically-motivated copy strategies. In addition, we employed a baseline sequence-to-sequence model (Dong and Lapata, 2016) which treats DRTSs as linearized trees.

For the document-level experiments, we built two baseline models. The first one treats documents as one long string (by concatenating all document sentences) and performs sentence-level parsing (DocSent). The second one parses each sentence in a document with a parser trained on the sentence-level version of the GMB and constructs a (flat) document tree by gathering all sentential DRTSs as children of a segmented DRS node (DocTree). We used the sentence-level DRTS parser for both baselines. We also compared four variants of our document-level model: one with multi-attention and shallow sentence representations (Shallow); one with multi-attention and deep sentence representations (Deep); a Deep model with supervised attention and alignments as features (DeepFeat); and finally, a Deep model with copying modulated by supervised attention (DeepCopy). All variants of our DRTS parser and comparison models adopt constraint-based inference.

Evaluation We evaluated the output of our semantic parser using COUNTER (van Noord et al., 2018a), a recently proposed metric suited to matching scoped meaning representations. COUNTER converts DRSs to sets of clauses and computes precision and recall on matching clauses. We transformed DRTSs to clauses as shown in Figure 5. b variables refer to DRS nodes, and children of DRS nodes correspond to clauses. We used a hill-climbing algorithm to match variables between predicted clauses and gold standard clauses. We report F_1 using exact match and partial match. For example, given predicted clauses "$b_0\ fall\ e_1,\ b_0\ Agent\ e_2\ x_1,\ b_0\ push\ e_2$" and gold standard clauses "$b_0\ fall\ e_1,\ b_0\ Agent\ e_1\ x_1$", exact F_1 is 0.4 ($1/3$ precision and $1/2$ recall) while partial F_1 is 0.67 ($4/7$ precision and $4/5$ recall).

6 Results

Parsing Sentences Table 2 summarizes results on the sentence-level semantic parsing task for our model (DRTS parser), Liu et al.'s (2018) model, and the sequence-to-sequence baseline (Seq2Seq). As can be seen, our system outperforms comparison models by a wide margin. The better performance over Liu et al. (2018) is due to the richer feature space we exploit and the application of linguistically-motivated copy strategies.

Parsing Documents Table 3 presents various ablation studies for the document-level model on the development set. Deep sentence representations when combined with multi-attention bring

[11] Models were trained on a single GPU without batches.

Models	par-F_1	exa-F_1
Seq2Seq	61.27	51.21
Liu et al. (2018)	74.31	68.72
DRTS parser	80.06	77.85

Table 2: Results (test set) on sentence-level GMB benchmark.

DRTS parser	par-F_1	exa-F_1
Shallow	66.63	61.74
Deep	71.01	65.42
DeepFeat	71.44	66.43
DeepCopy	75.89	69.45

Table 3: Results (dev set) on document-level GMB benchmark.

Models	par-F_1	exa-F_1
DocSent	57.10	53.27
DocTree	62.83	58.22
DeepCopy	70.83	66.56

Table 4: Results (test set) on document-level GMB benchmark.

DeepCopy	atomic	scoped	DRS	All
sentences	0.22	0.26	1.78	2.09
documents	3.57	4.54	25.02	30.75

Table 5: Percentage of ill-formed outputs without constraints during inference (test set); atomic refers to atomic nodes, scoped refers to scoped nodes and DRS referes to DRS nodes (from Section 2) violated.

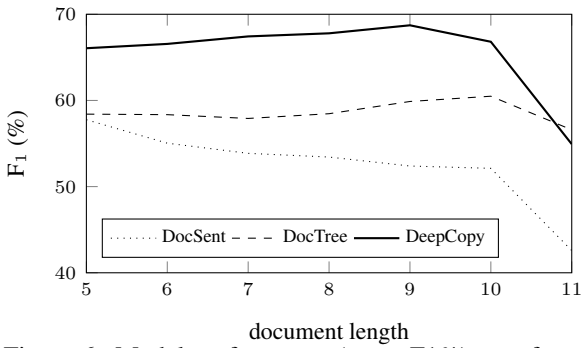

Figure 6: Model performance (exact F1%) as a function of document length (i.e., number of sentences).

improvements over shallow representations (+3.68 exact-F_1). Using alignments as features and as a way of highlighting where to copy from yields further performance gains both in terms of exact and partial F_1. The best performing variant is DeepCopy which combines supervised attention with copying. Table 4 shows our results on the test set (see the Appendix for an example of model output); we compare the best performing DRTS parser (DeepCopy) against two baselines which rely on our sentence-level parser (DocSent and DocTree). The DRTS parser, which has a global view of the document, outperforms variants which construct document representations by aggregating individually parsed sentences.

Influence of Constraints In Table 5, we examine whether constraint-based inference is helpful. In particular we show the percentage of ill-formed DRTSs when constraints are not enforced. We present results for the sentence- and document-level parsers overall and broken down according to the type of DRTS nodes being violated. 30.75% of document level DRTSs are ill-formed when constraints are not imposed during inference. This is in stark contrast with sentence-level outputs which are mostly well-formed (only 2.09% display violations of any kind). We observe that most violations concern elementary and segmented DRS nodes.

Influence of Document size Figure 6 shows how our parser (DeepCopy variant) and comparison systems perform on documents of varying length. Unsurprisingly, we observe that F_1 decreases with document length and that all systems have trouble modeling documents with 10 sentences and beyond. In general, DeepCopy has an advantage over comparison systems due to the more sophisticated alignment information and the fact that it aims to generate global document-level structures. Our results also indicate that modeling longer documents which are relatively few in the training set is challenging mainly because the parser cannot learn reliable representations for them. Moreover, as the size of documents increases, ambiguity for the resolution of coreferring expressions increases, suggesting that explicit modeling of anaphoric links might be necessary.

7 Related Work

Le and Zuidema (2012) were the first to train a data-driven DRT parser using a graph-based representation. Recently, Liu et al. (2018) conceptualized DRT parsing as a tree structure prediction problem which they modeled with a series of encoder-decoder architectures. van Noord et al. (2018b) adapt models from neural machine translation (Klein et al., 2017) to DRT parsing, also following a graph-based representation. Previous work has focused exclusively on sentences, whereas we design a general framework for parsing sentences *and* documents and provide a model which can be used interchangeably for both.

Various mechanisms have been proposed to improve sequence-to-sequence models including copying (Gu et al., 2016) and attention (Mikolov

et al., 2013). Our copying mechanism is more specialized and linguistically-motivated: it considers the semantics of the input text for deciding which tokens to copy. While our multi-attention mechanism is fairly general, it extracts features from different encoder representations (word- or sentence-level) and flexibly integrates supervised and unsupervised attention in a unified framework.

A few recent approaches focus on the alignment between semantic representations and input text, either as a preprocessing step (Foland and Martin, 2017; Damonte et al., 2017) or as a latent variable (Lyu and Titov, 2018). Instead, our parser implicitly models word-level alignments with multi-attention and explicitly obtains sentence-level alignments with supervised attention, aiming to jointly train a semantic parser.

8 Conclusions

In this work we proposed a novel semantic parsing task to obtain Discourse Representation Tree Structures and introduced a general framework for parsing texts of arbitrary length and granularity. Experimental results on two benchmarks show that our parser is able to obtain reasonably accurate sentence- and document-level discourse representation structures (77.85 and 66.56 exact-F_1, respectively). In the future, we would like to more faithfully capture the semantics of documents by explicitly modeling entities and their linking.

Acknowledgments We thank the anonymous reviewers for their feedback and Johan Bos for answering several questions relating to the GMB. We gratefully acknowledge the support of the European Research Council (Lapata, Liu; award number 681760), the EU H2020 project SUMMA (Cohen, Liu; grant agreement 688139) and Bloomberg (Cohen, Liu).

References

David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured decoding with doubly-recurrent neural networks. In *Proceedings of the 5th International Conference on Learning Representation (ICLR)*, Toulon, France.

Nicholas Asher. 1993. Reference to abstract objects in English: a philosophical semantics for natural language metaphysics. *Studies in Linguistics and Philosophy. Kluwer, Dordrecht*.

Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Diego, California.

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*, pages 178–186, Sofia, Bulgaria.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating discourse phenomena in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1304–1313, New Orleans, Louisiana.

Johan Bos, Valerio Basile, Kilian Evang, Noortje Venhuizen, and Johannes Bjerva. 2017. The groningen meaning bank. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, volume 2, pages 463–496. Springer.

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)*, pages 1215–1226, Vancouver, Canada.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 44–55, Vancouver, Canada.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, Christine Pao David Pallett, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: the atis-3 corpus. In *Proceedings of the workshop on ARPA Human Language Technology*, pages 43–48, Plainsboro, New Jersey.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pages 731–742, Melbourne, Australia.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. Cmu at semeval-2016 task 8: Graph-based amr parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206, San Diego, California.

William Foland and James H Martin. 2017. Abstract meaning representation parsing using lstm recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 463–472, Vancouver, Canada.

Naman Goyal and Jacob Eisenstein. 2016. A joint model of rhetorical discourse structure and summarization. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 25–34, Austin, Texas. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany.

Hans Kamp. 1981. A theory of truth and semantic representation. In J. A. G. Groenendijk, T. M. V. Janssen, and M. B. J. Stokhof, editors, *Formal Methods in the Study of Language*, volume 1, pages 277–322. Mathematisch Centrum, Amsterdam.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer, Dordrecht.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 1062–1068, Pittsburgh, Pennsylvania.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, Doha, Qatar.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, Banff, Canada.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada.

Tomáš Kočiskỳ, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087, Austin, Texas.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, Edinburgh, Scotland, UK.

Phong Le and Willem Zuidema. 2012. Learning compositional semantics for open domain semantic parsing. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 1535–1552, Mumbai, India.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon.

Jiangming Liu, Shay B. Cohen, and Mirella Lapata. 2018. Discourse representation structure parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 429–439, Melbourne, Australia.

Chunchuan Lyu and Ivan Titov. 2018. Amr parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407, Melbourne, Australia.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Rik van Noord, Lasha Abzianidze, Hessel Haagsma, and Johan Bos. 2018a. Evaluating scoped meaning representations. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Rik van Noord, Lasha Abzianidze, Antonio Toral, and Johan Bos. 2018b. Exploring neural methods for parsing discourse representation structures. *Transactions of the Association for Computational Linguistics*, 6:619–633.

Ella Rabinovich, Noam Ordan, and Shuly Wintner. 2017. Found in translation: Reconstructing phylogenetic language trees from translations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 530–540, Vancouver, Canada.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia.

Karin Sim Smith. 2017. On integrating discourse in machine translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 110–121, Copenhagen, Denmark.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the Deep Learning Workshop in the 31st International Conference on Machine Learning*, volume 37.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1050–1055, Portland, Oregon.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *PProceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 658–666, Edinburgh, Scotland, UK.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1416–1421, Denver, Colorado.

Bowei Zou, Guodong Zhou, and Qiaoming Zhu. 2014. Negation focus identification with contextual discourse information. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 522–530, Baltimore, Maryland. Association for Computational Linguistics.

A Constraint-based Inference

In this section we provide more formal detail on how our model applies constraint-based inference. In order to guide sequential predictions, we define a State Tracker (ST) equipped with four functions: INITIALIZATION initializes the ST, UPDATE updates the ST according to token y, ISTERMINATED determines whether the ST should terminate, and VALID returns the set of valid candidates in the current state. The state tracker provides an efficient interface for applying constraints during decoding. Sequential inference with the ST is shown in Algorithm 1; θ are model parameters and Y_k^{valid} all possible valid predictions at step k.

A.1 Stage 1

Algorithm 2 implements the ST functions for Stage 1; DRS denotes an elementary DRS node, SDRS denotes a segmented DRS node, *prop*SN is short for *proposition* scoped node, *segm*SN is short for *segment* scoped node, *simp*SN is short for *simple* scoped node. Function INITIALIZATION (lines 1–4) initializes the ST as an empty stack with a counter.

Lines 5–15 implement the function UPDATE, where y is placed on top of the stack if it is not a *CompletedSymbol* (lines 6–7) and the counter is incremented if y is an elementary DRS node (lines 8–9). The top of the stack is popped if y is a *CompletedSymbol* (line 12), i.e., the children of the node on top of the stack have been generated, and the stack is updated (line 13).

Lines 16–22 implement the function ISTERMINATED. If the stack is empty, decoding in Stage 1 is completed. Function ISTERMINATED is called after function UPDATE has been called at least once (see lines 7–8 in Algorithm 1).

Lines 23–63 implement the function VALID, which returns the set of valid candidates Y^{valid}

Algorithm 1 Inference with ST

```
1:  procedure INFERENCE(ST, θ)
2:      INITIALIZATION(ST)
3:      k = 0
4:      repeat
5:          Y_k^valid = VALID(ST)
6:          y_k^* = arg max_{y_k ∈ Y_k^valid} P(y_k | y_<k, θ)
7:          UPDATE(ST, y_k^*)
8:          k = k + 1
9:      until ISTERMINATED(ST)
10:     return [y_0^*, ..., y_{k-1}^*]
11: end procedure
```

in the current state. If the stack is empty, which means that a root of a DRTS should be constructed, Y^{valid} only includes elementary and segmented DRS nodes (lines 24–25). We use *top* to denote the top node of the stack (line 27). If *top* is a *proposition* scoped node or *segment* scoped node, Y^{valid} includes an elementary and segmented DRS node only if *top* has no children (lines 29–30), otherwise Y^{valid} includes *CompletedSymbol* (lines 31–32), showing that the scoped node should be completed with only one elementary or segmented DRS node as a child. The same constraints are applied to unary *simple* scoped nodes (lines 34–39). Similarly, binary *simple* scoped nodes should only have two elementary or segmented DRS nodes as children (lines 40–45).

If *top* is an elementary DRS node, Y^{valid} is initialized with the set $\{$*CompletedSymbol*$\}$ (line 47), because it can be completed without any child in Stage 1.[12] Furthermore, if the number of elementary DRS nodes is within the threshold MAX_DRS, *top* can have more children, i.e., Y^{valid} includes scoped nodes, except *segmented* scoped nodes (lines 48–49). If *top* is a segmented DRS node and has less than two children, Y^{valid} only includes segment scoped nodes (lines 52–53). Furthermore, if the number of elementary DRS nodes is within the threshold MAX_DRS, *top* can have more children, i.e., Y^{valid} includes *segmented* scoped nodes (lines 55–57).

A.2 Stage 2

The ST functions for Stage 2 are shown in Algorithm 3. Lines 1–5 implement the function INITIALIZATION, which initializes ST as a relation counter, a type flag, and a completed flag. The relation counter records the number of relations that have been already constructed. The type flag shows the type of nodes, i.e., e for elementary DRS nodes or s for segmented DRS nodes, based on which the relations are constructed. The completed flag checks if the construction is completed. Lines 6–11 implement the function UPDATE. If *CompletedSymbol* is predicted, the completed flag is set to true, and the completed flag is checked (lines 12–14, function ISTERMINATED).

Lines 15–24 implement the function VALID. If the number of constructed relations is zero, Y^{valid} only includes R (lines 16–17). If the number of constructed relations is within the threshold $\text{MAX\_REL}_{\text{ST.type}}$, it is possible to construct more relations (lines 18–19). If the number of children exceeds the threshold, Y^{valid} only includes *CompletedSymbol* to complete the construction of relations (lines 20–21).

A.3 Stage 3

Algorithm 4 implements the ST functions for Stage 3, where V_e includes entity variables, event variables, state variables, time variables, proposition variables, and constants, and V_s includes segment variables. Lines 1–5 (INITIALIZATION) initialize the ST with a variable counter, a type flag, and a completed flag. The variable counter records the number of variables that have already been constructed. The type flag shows the type of nodes (e for elementary DRS nodes or s for segmented DRS nodes), based on which the variables are constructed. The completed flag checks if the construction is completed. Lines 6–11 implement the function UPDATE. If *CompletedSymbol* is predicted, the completed flag is set to true and checked (lines 12–14, function ISTERMINATED).

Lines 15–28 implement the function VALID. If no variables are constructed, Y^{valid} only includes $V_{\text{ST.type}}$ (lines 16–17). If only one variable is constructed and ST.type is a segmented DRS, Y^{valid} only includes V_s to construct one more variable because relations in segmented DRS nodes are binary (lines 21–22). If two variables are constructed, Y^{valid} only includes *CompletedSymbol* (line 25). Note that indices of variables are in increased order.

B Example Output

We provide example output of our model (DRTS parser, DeepCopy variant) for the GMB document below in Figure 7.

European Union energy officials will

[12] Atomic nodes are constructed in Stage 2.

hold an emergency meeting next week amid concerns that the Russian-Ukrainian dispute over natural gas prices could affect EU gas supplies. An EU statement released Friday says the meeting is aimed at finding a common approach. It also expresses the European Commission's concern about the situation, but says the EU top executive body remains confident an agreement will be reached. A Russian cut-off of supplies to Ukraine will reduce the amount of natural gas flowing through the main pipeline toward Europe. But the commission says there is no risk of a gas shortage in the short term. German officials say they are hoping for a quick resolution to the dispute. Government spokesman, Ulrich Wilhelm says officials have been in contact with both sides at a working level, but will not mediate.

Algorithm 2 State Tracker for Stage 1

```
procedure INITIALIZATION(ST)
    ST.stack = []
    ST.count = 0
end procedure
procedure UPDATE(ST, y)
    if y is not CompletedSymbol then
        ST.stack.push(y)
        if y is DRS then
            ST.count += 1
        end if
    else
        ST.stack.pop()
        ST.stack.top.childnum += 1
    end if
end procedure
procedure ISTERMINATED(ST)
    if ST.stack.empty() then
        return True
    else
        return False
    end if
end procedure
procedure VAILD(ST)
    if ST.stack.empty() then
        Y^Valid = {DRS, SDRS}
    else
        top = ST.stack.top
        if top is propSN or segmSN then
            if top.childnum = 0 then
                Y^Valid = {DRS, SDRS}
            else
                Y^Valid = {CompletedSymbol}
            end if
        else if top is unary simpSN then
            if top.childnum = 0 then
                Y^Valid = {DRS, SDRS}
            else
                Y^Valid = {CompletedSymbol}
            end if
        else if top is binary simpSN then
            if top.childnum ≤ 1 then
                Y^Valid = {DRS, SDRS}
            else
                Y^Valid = {CompletedSymbol}
            end if
        else if top is DRS then
            Y^Valid = {CompletedSymbol}
            if ST.count < MAX_DRS then
                Y^Valid = Y^Valid ∪ {propSN, simpSN}
            end if
        else if top is SDRS then
            if top.childnum < 2 then
                Y^Valid = {segmSN}
            else
                Y^Valid = {CompletedSymbol}
                if ST.count < MAX_DRS then
                    Y^Valid = Y^Valid ∪ {segmSN}
                end if
            end if
        end if
    end if
    return Y^Valid
end procedure
```

Algorithm 3 State Tracker for Stage 2

procedure INITIALIZATION(ST, type)
    ST.count = 0
    ST.completed = False
    ST.type = type
end procedure
procedure UPDATE(ST, y)
    ST.count += 1
    if y is *CompletedSymbol* **then**
        ST.completed = True
    end if
end procedure
procedure ISTERMINATED(ST)
    return ST.completed
end procedure
procedure VALID(ST)
    if ST.count $= 0$ **then**
        $Y^{\text{Valid}} = R_{\text{ST.type}}$
    else if ST.count $<$ MAX_REL$_{\text{ST.type}}$ **then**
        $Y^{\text{Valid}} = R_{\text{ST.type}} \cup \{\textit{CompletedSymbol}\}$
    else
        $Y^{\text{Valid}} = \{\textit{CompletedSymbol}\}$
    end if
    return Y^{Valid}
end procedure

Algorithm 4 State Tracker for Stage 3

procedure INITIALIZATION(ST, type)
    ST.count = 0
    ST.completed = False
    ST.type = type
end procedure
procedure UPDATE(ST, y)
    ST.count += 1
    if y is *CompletedSymbol* **then**
        ST.completed = True
    end if
end procedure
procedure ISTERMINATED(ST)
    return ST.completed
end procedure
procedure VALID(ST)
    if ST.count $= 0$ **then**
        $Y^{\text{Valid}} = V_{\text{ST.type}}$
    else if ST.count $= 1$ **then**
        if ST.type is elementary DRS **then**
            $Y^{\text{Valid}} = V_{\text{ST.type}} \cup \{\textit{CompletedSymbol}\}$
        else if ST.type is segmented DRS **then**
            $Y^{\text{Valid}} = V_{\text{ST.type}}$
        end if
    else
        $Y^{\text{Valid}} = \{\textit{CompletedSymbol}\}$
    end if
    return Y^{Valid}
end procedure

Figure 7: Output of DRTS parser (DeepCopy variant) for the document in Section 2.