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; Kočiský 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).

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., $\max(x_1)$, $\text{male}(x_1)$) 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-
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. 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 answering (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 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 $k$.

$R$ 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 duplication, 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 $C$. 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-

---

2https://github.com/LeonCrashCode/TreeDRSparsing
3Segment variables originate from Segmented Discourse Representation Theory (SDRT; Asher and Lascarides 2003), and denote units connected by discourse relations.
4Duplex represents wh-questions (e.g., who, what, how).
5In our formulation, the only constants used are for denoting numbers. Proper names are denoted by relations, such as 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 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 $\langle d \rangle_{w00}, ..., \langle sep_i \rangle_{wij}, ..., \langle /d \rangle_{wkn}$, 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 $i$th sentence.\(^\text{5}\)

The $j$th token in the $i$th sentence of a document is represented by vector $x_{ij} = f([e_{w_{ij}}; e_{w_{ij}}; e_{\ell_{ij}}])$ 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):

$$\overrightarrow{h_{x00}} : \overrightarrow{h_{xmn}} = \text{biLSTM}(x_{00} : x_{mn}),$$

where $\overrightarrow{h_{xij}}$ denotes the hidden representation of the encoder for $x_{ij}$, which denotes the input representation of token $j$ in sentence $i$.

\(^\text{5}\)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 $\langle /d \rangle$.\)
Deep Sentence Representation An alternative to the shallow sentence representation just described, is a biLSTM encoder:

\[
\begin{align*}
\tilde{h}_{x_0} \rightarrow \tilde{h}_{x_m} &= \text{biLSTM}(h_{x_0} : h_{x_m}),
\end{align*}
\]

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 \( v \). Given encoder representations \( H_x = [h_{x_0}, h_{x_1}, \ldots, h_{x_m}] \), we extract features by applying a standard attention mechanism (Bahdanau et al., 2015) on encoder representations \( h_{y_k}^v \):

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

where weight \( \beta_{ki}^v \) is computed by:

\[
\beta_{ki}^v = \frac{\exp(h_{y_k}^v h_{x_i})}{\sum_{o} \exp(h_{y_k}^v h_{x_o})}.
\]

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 non-terminals \( y_{0t}^s, \ldots, y_{kt}^s \) (see Figure 2). The probabilistic distribution of the \( k \)th prediction is:

\[
P(y_{kt}^s | y_{kt-1}^s, H_x) = \text{SoftMax}(s_{kt}^s),
\]

where \( H_x \) refers to the encoder representations and score \( s_{kt}^s \) is computed as:

\[
s_{kt}^s = f([h_{y_k}^w; \text{Attn}^w(h_{y_k}^w, [h_{x_0} ; h_{x_m}]) ; \text{Attn}^s(h_{y_k}^s, [h_{x_0} ; h_{x_m}])]),
\]

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

Stage 2 Given elementary or segmented DRS nodes generated in Stage 1, atomic nodes \( y_{0d}^n, \ldots, y_{kd}^n \) are predicted (see Figure 2), with the aid of copy strategies which we discuss shortly. The probabilistic distribution of the \( k \)th prediction is:

\[
P(y_{kd}^n | y_{kt}^s, H_x, H_y^s) = \text{SoftMax}([s_{kd}^s : s_{kd}^c]),
\]

where \( s_{kd}^s \) and \( s_{kd}^c \) are generation and copy scores, respectively, over the \( k \)th prediction:

\[
s_{kd}^s = f([h_{y_k}^w ; \text{Attn}^w(h_{y_k}^w, [h_{x_0} ; h_{x_m}]) ; \text{Attn}^s(h_{y_k}^s, [h_{x_0} ; h_{x_m}])])
\]

\[
s_{kd}^c = f([h_{y_k}^c ; \text{Attn}^c(h_{y_k}^c, [h_{x_0} ; h_{x_m}]) ; \text{Attn}^s(h_{y_k}^s, [h_{x_0} ; h_{x_m}])]).
\]
where \( [h_{\ell_0}^{\text{copy}} : h_{\ell_2}^{\text{copy}}] \) are copy representations used for copy scoring; and \( h_{\ell_0}^{\text{copy}} \) is the hidden representation of the decoder in Stage 2, which is obtained based on how the previous token was constructed:

\[
h_{\ell_0}^{\text{nd}} = \begin{cases} \text{LSTM}(e_{\ell_0}^{\text{copy}}(h_{\ell_0}^{\text{copy}})) & y_{\ell_0}^{\text{nd}} \text{ is copied} \\ \text{LSTM}(e_{\ell_0}^{\text{atm}}) & y_{\ell_0}^{\text{nd}} \text{ is generated} \\ \text{LSTM}(g_{\text{atm2nd}}(h_{\text{drs}})) & k = 0 \end{cases}
\]

The generation of atomic nodes in the second stage is conditioned on \( h_{\text{drs}} \), the decoder hidden representation of elementary or segmented DRS nodes from Stage 1 by the linear function \( g_{\text{atm2nd}} \).

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(x1), eat(e1)), 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(x1) and company(x2) will be copied from oil and company, while node of(x2, x1) will not be copied from deprived of.

Copy representations \( M_d = [h_{\ell_0}^{\text{copy}} : h_{\ell_2}^{\text{copy}}] \) are constructed for each document \( d \) from its encoder hidden representations \( [h_{x0} : h_{xmn}] \), 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 \( d \):

\[
h_{\ell_2}' = \frac{1}{N} \sum_{(ij) : \ell_{ij} = \ell_2'} h_{x_{ij}},
\]

and \( N \) is the number of tokens with lemma \( \ell_2' \).

**Stage 3** Finally, we generate terminals, i.e., atomic node variables \( y_{0d}^{\text{nd}}, \ldots, y_{k}^{\text{nd}} \) (see Figure 2).

The probabilistic distribution of the \( k \)th prediction is:

\[
P(y_{k}^{\text{nd}} | y_{<k}^{\text{nd}}, H_x, H_{yd}) = \text{SoftMax}(s_{k}^{\text{rd}}),
\]

\[
s_{k}^{\text{rd}} = f([h_{y_{k}^{\text{rd}}}; \text{Attn}_{\text{word}}(h_{y_{k}^{\text{rd}}}, [h_{x0}; h_{xmn}])); \text{Attn}_{\text{sent}}(h_{y_{k}^{\text{rd}}}, [h_{x0}; h_{xmn}]))
\]

where \( h_{y_{k}^{\text{rd}}} \) is the decoder hidden representation in the third stage:

\[
h_{y_{k}^{\text{rd}}} = \begin{cases} \text{LSTM}(e_{y_{k}^{\text{rd}}}) & k \neq 0 \\ \text{LSTM}(g_{\text{atm2nd}}(h_{\text{atm}})) & k = 0 \end{cases}
\]

Here, the generation of variables is conditioned upon \( h_{\text{atm}} \), the decoder hidden representation of atomic nodes from the second stage by the linear function \( g_{\text{atm2nd}} \).

### 3.3 Training

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

\[
L(\theta) = -\frac{1}{N} \sum_j \log p_j,
\]

where \( p_j \) is the distribution of output tokens, \( \theta \) 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:

\[
\text{Score}_{\text{align}}(h_{y_k}, [h_{x0}; h_{x_m}]) = [\beta_{k0}^{\text{align}} : \beta_{km}^{\text{align}}]
\]
where \( p_{k}^{\text{align}} = P(a_k = m|h_{y_k}, [h_{x_0} : h_{x_m}]) \), i.e., the probabilistic distribution over alignments from sentences to the \( k \)th prediction in the decoder, where \( a_k = m \) denotes the \( k \)th prediction aligned to the \( m \)th 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 \( p_k^{\text{align}} \) 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:

\[
s_{stg}^k = f(h_{y_{stg}}; \text{Attn}^{\text{word}}(h_{y_{stg}}, [h_{x_0} : h_{x_m}]), h_{x_{stg}}; \text{Attn}^{\text{sent}}(h_{y_{stg}}, [h_{x_0} : h_{x_m}]),)
\]

where \( stg \in \{ \text{st}, \text{nd}, \text{rd} \} \), and \( h_{x_{stg}} \) is the \( a_k \)th 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 \( M_i = [h_{i_{0}}^\text{copy} : h_{i_{1}}^\text{copy}], L'_i \subseteq L_i \) and \( L'_i \) is the set of distinct lemmas in the \( i \)th sentence:

\[
h_{i_{j}}^\text{copy} = \frac{1}{N} \sum_{(ij):\ell_{ij}=\ell'_i} h_{x_{ij}},
\]

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

\[
s_{k}^\text{copy} = \text{Score}^\text{copy}(h_{y_{stg}}, M_a)
\]

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.

---

**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 \( Y_k^\text{valid} \) 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 \( \theta \) are the parameters of the model, and \( Y_k^\text{valid} \) 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 \( R_s \) 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.
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.\footnote{Models were trained on a single GPU without batches.}

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 “$b_0 \text{ fall } e_1$, $b_0 \text{ Agent } e_2 x_1$, $b_0 \text{ push } e_2$” and gold standard clauses “$b_0 \text{ fall } e_1$, $b_0 \text{ Agent } e_1 x_1$”, 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

<table>
<thead>
<tr>
<th>Sentences</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>#sent</td>
<td>avg$_{w}$</td>
</tr>
<tr>
<td>train</td>
<td>41,563</td>
</tr>
<tr>
<td>dev</td>
<td>5,173</td>
</tr>
<tr>
<td>test</td>
<td>5,451</td>
</tr>
</tbody>
</table>

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).
improvements over shallow representations (+3.68 exact-F1). 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 F1. 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.

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 refers to DRS nodes (from Section 2) violated.

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 F1 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-F1, 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


Naman Goyal and Jacob Eisenstein. 2016. A joint
William Foland and James H Martin. 2017. Abstract
Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime

William Foland and James H Martin. 2017. Abstract


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 \( y \), IS-TERMINATED 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; \( \theta \) are model parameters and \( Y^\text{valid}_k \) 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, \( \text{propSN} \) is short for \( \text{proposition} \) scoped node, \( \text{segmSN} \) is short for \( \text{segment} \) scoped node, \( \text{simpSN} \) is short for \( \text{simple} \) 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 \( Y^\text{valid} \).
Algorithm 1 Inference with ST

1: procedure INFEERENCE(ST, \( \theta \))
2: INITIALIZATION(ST)
3: \( k = 0 \)
4: repeat
5: \( Y_k^{\text{valid}} = \text{VALID}(ST) \)
6: \( y_k^* = \arg \max_{y_k \in Y_k^{\text{valid}}} P(y_k | y_{\leq k}, \theta) \)
7: UPDATE(ST, \( y_k^* \))
8: \( k = k + 1 \)
9: until ISTERMINATED(ST)
10: return \([y_1^*, ... , y_{k-1}^*]\)
11: end procedure

in the current state. If the stack is empty, which means that a root of a DRTS should be constructed, \( Y^{\text{valid}} \) 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, \( Y^{\text{valid}} \) includes an elementary and segmented DRS node only if \( top \) has no children (lines 29–30), otherwise \( Y^{\text{valid}} \) 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, \( Y^{\text{valid}} \) 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, \( top \) can have more children, i.e., \( Y^{\text{valid}} \) includes scoped nodes, except segmented scoped nodes (lines 48–49). If \( top \) is a segmented DRS node and has less than two children, \( Y^{\text{valid}} \) only includes segment scoped nodes (lines 52–53). Furthermore, if the number of elementary DRS nodes is within the threshold MAX, \( top \) can have more children, i.e., \( Y^{\text{valid}} \) 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 already been 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, \( Y^{\text{valid}} \) only includes \( R \) (lines 16–17). If the number of constructed relations is within the threshold MAX, 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, \( Y^{\text{valid}} \) only includes \( V_{\text{ST, type}} \) (lines 16–17). If only one variable is constructed and ST.type is a segmented DRS, \( Y^{\text{valid}} \) 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, \( Y^{\text{valid}} \) 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
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

```plaintext
1: procedure INITIALIZATION(ST)
2:     ST.stack = []
3:     ST.count = 0
4: end procedure
5: procedure UPDATE(ST, y)
6:     if y is not CompletedSymbol then
7:         ST.stack.push(y)
8:     if y is DRS then
9:         ST.count += 1
10:     else
11:         ST.stack.pop()
12:         ST.stack.top.childnum += 1
13: end if
14: end procedure
15: procedure IS_TERMINATED(ST)
16:     if ST.stack.empty() then
17:         return True
18:     else
19:         return False
20: end if
21: end procedure
22: procedure VALID(ST)
23:     if ST.stack.empty() then
24:         YValid = {DRS, SDRS}
25:     else
26:         top = ST.stack.top
27:         if top is propSN or segmSN then
28:             if top.childnum = 0 then
29:                 YValid = {DRS, SDRS}
30:             else
31:                 YValid = {CompletedSymbol}
32:             end if
33:         else if top is unary simpSN then
34:             if top.childnum = 0 then
35:                 YValid = {DRS, SDRS}
36:             else
37:                 YValid = {CompletedSymbol}
38:             end if
39:         else if top is binary simpSN then
40:             if top.childnum ≤ 1 then
41:                 YValid = {DRS, SDRS}
42:             else
43:                 YValid = {CompletedSymbol}
44:             end if
45:         else if top is DRS then
46:             YValid = {CompletedSymbol}
47:             if ST.count < MAX_DRS then
48:                 YValid = YValid ∪ {propSN, simpSN}
49:             end if
50:         else if top is SDRS then
51:             if top.childnum < 2 then
52:                 YValid = {segmSN}
53:             else
54:                 YValid = {CompletedSymbol}
55:             end if
56:         else
57:             YValid = {CompletedSymbol}
58:             if ST.count < MAX_DRS then
59:                 YValid = YValid ∪ {segmSN}
60:         end if
61:     end if
62: return YValid
63: end procedure
```
Algorithm 3 State Tracker for Stage 2

1: procedure INITIALIZATION(ST, type)
2:   ST.count = 0
3:   ST.completed = False
4:   ST.type = type
5: end procedure
6: procedure UPDATE(ST, y)
7:   ST.count += 1
8:   if y is CompletedSymbol then
9:     ST.completed = True
10: end if
11: end procedure
12: procedure ISTERMINATED(ST)
13:   return ST.completed
14: end procedure
15: procedure VALID(ST)
16:   if ST.count = 0 then
17:     YValid = RST.type
18:   else if ST.count < MAX_RELST.type then
19:     YValid = RST.type ∪ {CompletedSymbol}
20:   else
21:     YValid = {CompletedSymbol}
22:   end if
23:   return YValid
24: end procedure

Algorithm 4 State Tracker for Stage 3

1: procedure INITIALIZATION(ST, type)
2:   ST.count = 0
3:   ST.completed = False
4:   ST.type = type
5: end procedure
6: procedure UPDATE(ST, y)
7:   ST.count += 1
8:   if y is CompletedSymbol then
9:     ST.completed = True
10: end if
11: end procedure
12: procedure ISTERMINATED(ST)
13:   return ST.completed
14: end procedure
15: procedure VALID(ST)
16:   if ST.count = 0 then
17:     YValid = VST.type
18: else if ST.count = 1 then
19:   if ST.type is elementary DRS then
20:     YValid = VSType ∪ {CompletedSymbol}
21:   else if ST.type is segmented DRS then
22:     YValid = VSType
23: end if
24: else
25:   YValid = {CompletedSymbol}
26: end if
27: return YValid
28: end procedure
Figure 7: Output of DRTS parser (DeepCopy variant) for the document in Section 2.