Broad-Coverage Semantic Parsing as Transduction

Sheng Zhang **Xutai Ma** **Kevin Duh** **Benjamin Van Durme**
Johns Hopkins University
{zsheng2, xutai_ma}@jhu.edu
{kevinduh, vandurme}@cs.jhu.edu

Abstract

We unify different broad-coverage semantic parsing tasks under a transduction paradigm, and propose an attention-based neural framework that *incrementally* builds a meaning representation via a sequence of semantic relations. By leveraging multiple attention mechanisms, the transducer can be effectively trained without relying on a pre-trained aligner. Experiments conducted on three separate broad-coverage semantic parsing tasks – AMR, SDP and UCCA – demonstrate that our attention-based neural transducer improves the state of the art on both AMR and UCCA, and is competitive with the state of the art on SDP.

1 Introduction

Broad-coverage semantic parsing aims at mapping *any* natural language text, regardless of its domain, genre, or even the language itself, into a general-purpose meaning representation. As a long-standing topic of interest in computational linguistics, broad-coverage semantic parsing has targeted a number of meaning representation frameworks, including CCG (Steedman, 1996, 2001), DRS (Kamp and Reyle, 1993; Bos, 2008), AMR (Banarescu et al., 2013), UCCA (Abend and Rappoport, 2013), SDP (Oepen et al., 2014, 2015), and UDS (White et al., 2016).[1] Each of these frameworks has their specific formal and linguistic assumptions. Such framework-specific "*balkanization*" results in a variety of framework-specific parsing approaches, and the state-of-the-art semantic parser for one framework is not always applicable to another. For instance, the state-of-the-art approaches to SDP parsing (Dozat and Manning, 2018; Peng et al., 2017a) are not directly transferable to AMR and UCCA because of the lack of explicit alignments between tokens in the sentence and nodes in the semantic graph.

While transition-based approaches are adaptable to different broad-coverage semantic parsing tasks (Wang et al., 2018; Hershcovich et al., 2018; Damonte et al., 2017), when it comes to representations such as AMR whose nodes are *unanchored* to tokens in the sentence, a pre-trained aligner has to be used to produce the reference transition sequences (Wang et al., 2015; Damonte et al., 2017; Peng et al., 2017b). In contrast, there are attempts to develop attention-based approaches in a graph-based parsing paradigm (Dozat and Manning, 2018; Zhang et al., 2019), but they lack parsing incrementality, which is advocated in terms of computational efficiency and cognitive modeling (Nivre, 2004; Huang and Sagae, 2010).

In this paper, we approach different broad-coverage semantic parsing tasks under a unified framework of transduction. We propose an attention-based neural transducer that extends the two-stage semantic parser of Zhang et al. (2019) to directly transduce input text into a meaning representation in *one* stage. This transducer has properties of both transition-based approaches and graph-based approaches: on the one hand, it builds a meaning representation incrementally via a sequence of semantic relations, similar to a transition-based parser; on the other hand, it leverages multiple attention mechanisms used in recent graph-based parsers, thereby removing the need for pre-trained aligners.

Requiring only minor task-specific adaptations, we apply this framework to three separate broad-coverage semantic parsing tasks: AMR, SDP, and UCCA. Experimental results show that our neural transducer outperforms the state-of-the-art parsers on AMR (77.0% F1 on LDC2017T10 and 71.3%

[1]Abbreviations respectively denote: Combinatory Categorical Grammar, Discourse Representation Theory, Abstract Meaning Representation, Universal Conceptual Cognitive Annotation, Semantic Dependency Parsing, and Universal Decompositional Semantics.

F1 on LDC2014T12) and UCCA (76.6% F1 on the English-Wiki dataset v1.2), and is competitive with the state of the art on SDP (92.2% F1 on the English DELPH-IN MRS dataset).

2 Background and Related Work

We provide summary background on the meaning representations we target, and review related work on parsing for each.

Abstract Meaning Representation (AMR; Banarescu et al., 2013) encodes sentence-level semantics, such as predicate-argument information, reentrancies, named entities, negation and modality, into a rooted, directed, and usually acyclic graph with node and edge labels. AMR graphs abstract away from syntactic realizations, i.e., there is no explicit correspondence between elements of the graph and the surface utterance. Fig. 1(a) shows an example AMR graph.

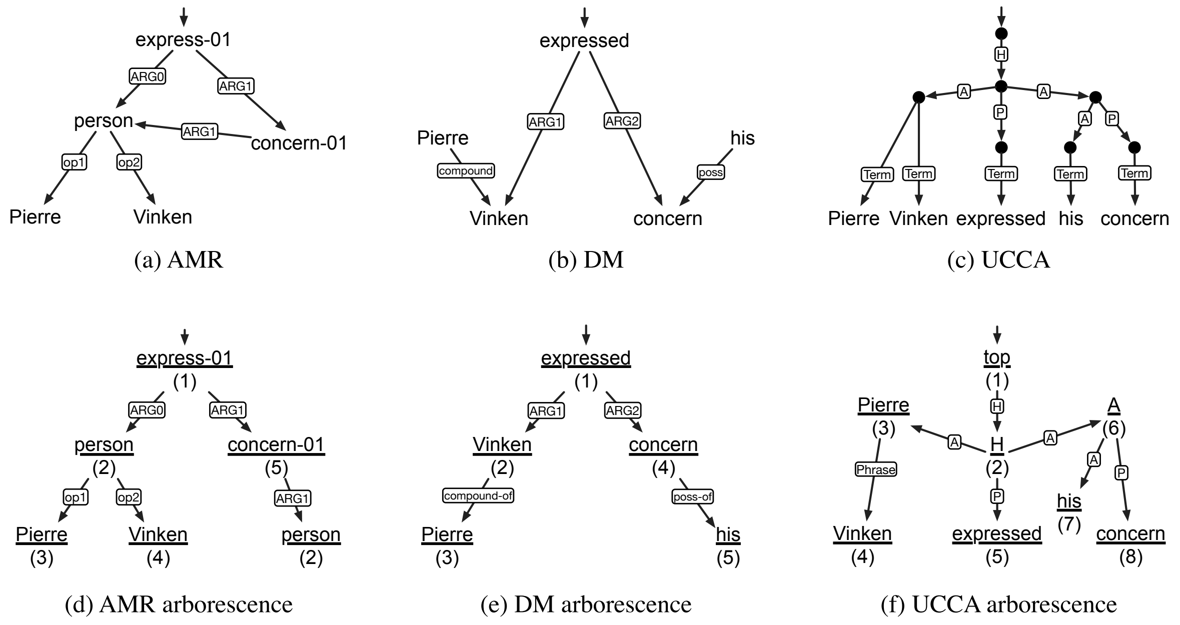
Since its first general release in 2014, AMR has been a popular target of data-driven semantic parsing, notably in two SemEval shared tasks (May, 2016; May and Priyadarshi, 2017). Graph-based parsers build AMRs by identifying concepts and scoring edges between them, either in a pipeline (Flanigan et al., 2014), or jointly (Zhou et al., 2016; Lyu and Titov, 2018; Zhang et al., 2019). This two-stage parsing process limits the parser incrementality. Transition-based parsers either transform dependency trees into AMRs (Wang et al., 2015, 2016; Goodman et al., 2016), or employ transition systems specifically tailored to AMR parsing (Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017). Transition-based parsers rely on pre-trained aligner produce the reference transitions. Grammar-based parsers leverage external semantic resources to derive AMRs compositionally based on CCG rules (Artzi et al., 2015), or SHRG rules (Peng et al., 2015). Another line of work uses neural model translation models to convert sentences into *linearized* AMRs (Barzdins and Gosko, 2016; Peng et al., 2017b), but has relied on data augmentation to produce effective parsers (van Noord and Bos, 2017; Konstas et al., 2017). Our parser differs from the previous ones in that it has incrementality without relying on pre-trained aligners, and can be effectively trained without data augmentation.

Semantic Dependency Parsing (SDP) was introduced in 2014 and 2015 SemEval shared tasks (Oepen et al., 2014, 2015). It is centered around three semantic formalisms – DM (DELPH-IN MRS; Flickinger et al., 2012; Oepen and Lønning, 2006), PAS (Predicate-Argument Structures; Miyao and Tsujii, 2004), and PSD (Prague Semantic Dependencies; Hajič et al., 2012) – representing predicate-argument relations between content words in a sentence. Their annotations have been converted into bi-lexical dependencies, forming directed graphs whose nodes injectively correspond to surface lexical units, and edges represent semantic relations between nodes. In this work, we focus on only the DM formalism. Fig. 1(b) shows an example DM graph.

Most recent parsers for SDP are graph-based: Peng et al. (2017a, 2018) use a max-margin classifier on top of a BiLSTM, with the factored score for each graph over predicates, unlabeled arcs, and arc labels. Multi-task learning approaches and disjoint data have been used to improve the parser performance. Dozat and Manning (2018) extend an LSTM-based syntactic dependency parser to produce graph-structured dependencies, and carefully tune it to state of the art performance. Wang et al. (2018) extend the transition system of Choi and McCallum (2013) to produce non-projective trees, and use improved versions of stack-LSTMs (Dyer et al., 2015) to learn representation for key components. All of these are specialized for bi-lexical dependency parsing, whereas our parser can effectively produce both bi-lexical semantics graphs, and graphs that are less anchored to the surface utterance.

Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013) targets a level of semantic granularity that abstracts away from syntactic paraphrases in a typologically-motivated, cross-linguistic fashion. Sentence representations in UCCA are directed acyclic graphs (DAG), where terminal nodes correspond to surface lexical tokens, and non-terminal nodes to semantic units that participate in super-ordinate relations. Edges are labeled, indicating the role of a child in the relation the parent represents. Fig. 1(c) shows an example UCCA DAG.

The first UCCA parser is proposed by Hershcovich et al. (2017), where they extend a transition system to produce DAGs. To leverage other semantic resources, Hershcovich et al. (2018) is one of the few attempts to present (lossy) conversion from AMR, SDP and Universal Dependencies (UD; Nivre et al., 2016) to a unified UCCA-



(a) AMR (b) DM (c) UCCA

(d) AMR arborescence (e) DM arborescence (f) UCCA arborescence

Figure 1: Meaning representation in the task-specific format – (a) AMR, (b) DM, and (c) UCCA – for an example sentence "*Pierre Vinken expressed his concern*". Meaning representation (d), (e) and (f) are in the unified arborescence format, which are converted from (a), (b) and (c) respectively.

based DAG format. They explore multi-task learning under the unified format. While multi-task learning improves UCCA parsing results, it shows poor performance on AMR, SDP and UD parsing. In contrast, different semantic parsing tasks are formalized in our unified transduction paradigm with no loss, and our approach achieves state-of-the-art or competitive performance on each task, using only single-task data.

3 Unified Transduction Problem

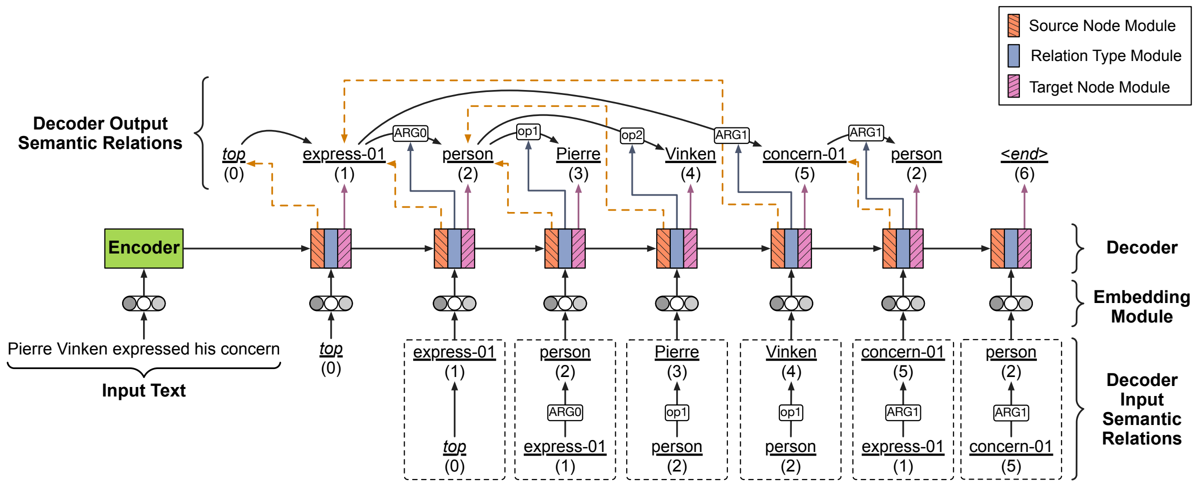
3.1 Unified Arborescence Format

We first introduce a unified target format for different broad-coverage semantic parsing tasks. Meaning representation in the unified format is an arborescence (aka, a directed rooted tree), which is converted from its corresponding task-specific semantic graph via the following *reversible* steps:

AMR Reentrancy is what can make an AMR graph not an arborescence (it introduces cycles). Following Zhang et al. (2019), we convert an AMR graph into an arborescence by duplicating nodes that have reentrant relations; that is, whenever a node has a reentrant relation, we make a copy of that node and use the copy to participate in the relation, thereby resulting in an arborescence. Next, in order to preserve the reentrancy information, we assign a node index to each node. Duplicated nodes are assigned the same index as the original node. Fig. 1(d) shows an AMR arborescence converted from Fig. 1(a): two "*person*" nodes have the same node index 2. The original AMR graph can be recovered by merging identically indexed nodes.

DM We first break the DM graph into a set of *weakly* connected subgraphs. For each subgraph, if it has the *top* node, we treat *top* as root; otherwise, we treat the node with the max outdegree as root. We then run depth-first traversal over each subgraph from its root to yield an arborescence, and repeat the following three steps until no more edges can be added to the arborescence: (1) we run breadth-first traversal over the arborescence from the root until we find a node that has an incoming edge not belonging to the arborescence; (2) we reverse the edge and add a `-of` suffix to the edge label; (3) we run depth-first search from that node to include more edges to the arborescence. During the whole process, we add node indices and duplicate reentrant nodes in the same way as AMR conversion. Finally, we connect arborescences by adding a `null` edge from *top* to other arborescence roots. Fig. 1(e) shows a DM arborescence converted from Fig. 1(b). The original DM graph can be recovered by removing `null` edges, merging identically indexed nodes, and reversing edges with `-of` suffix.

UCCA To date, official UCCA evaluation only considers UCCA's *foundational* layer, which is al-



Source Node Module
Relation Type Module
Target Node Module
Decoder Output Semantic Relations
top (0)
express-01 (1)
person (2)
Pierre (3)
Vinken (4)
concern-01 (5)
person (2)
<end> (6)
ARG0
op1
op2
ARG1
ARG1
Encoder
Decoder
Embedding Module
Pierre Vinken expressed his concern
Input Text
top (0)
express-01 (1) top (0)
person (2) ARG0 express-01 (1)
Pierre (3) op1 person (2)
Vinken (4) op1 person (2)
concern-01 (5) ARG1 express-01 (1)
person (2) ARG1 concern-01 (5)
Decoder Input Semantic Relations

Figure 2: The encoder-decoder architecture of our attention-based neural transducer. An encoder encodes the input text into hidden states. A decoder is composed by three modules: a target node module, a relation type module, and a source node module. At each decoding time step, the decoder takes the previous semantic relation as input, and outputs a new semantic relation in a *factorized* way: firstly, the target node module produces a new target node; secondly, the source node module *points* to a preceding node as a new source node; finally, the relation type module predicts the relation type between source and target nodes.

ready an arborescence. We convert it to the unified arborescence format by first collapsing subgraphs of pre-terminal nodes: we replace each pre-terminal node with its first terminal node; if the pre-terminal node has other terminals, we add a special `phrase` edge from the first terminal node to other terminal nodes. The collapsing step largely reduces the number of terminal nodes in UCCA. We then add labels to the remaining non-terminal nodes. Each node label is simply the same as its incoming edge label. We find that adding node labels improves performance of our neural transducer (See Section 6.2 for the experimental results). Lastly, we add node indices in the same way as AMR conversion. Fig. 1(f) shows a DM arborescence converted from Fig. 1(c). The original UCCA DAG can be recovered by expanding pre-terminal subgraphs, and removing non-terminal node labels.

3.2 Problem Formalization

For any broad-coverage semantic parsing task, we denote the input text by X, and the output meaning representation in the unified arborescence format by Y, where X is a sequence of tokens $\langle x_1, x_2, ..., x_n\rangle$ and Y can be decomposed as a sequence of semantic relations $\langle y_1, y_2, ..., y_m\rangle$. A relation y is a tuple $\langle u, d^u, r, v, d^v\rangle$, consisting of a source node label u, a source node index d^u, a relation type r, a target node label v, and a target node index d^v.

Let $\mathcal{Y}$ be the *output space*. The unified transduction problem is to seek the most-likely sequence of semantic relations $\hat{Y}$ given X:

$$\begin{aligned}\hat{Y} &= \arg\max_{Y\in\mathcal{Y}} \mathrm{P}(Y \mid X)\\ &= \arg\max_{Y\in\mathcal{Y}} \prod_{i}^{m} \mathrm{P}(y_i \mid y_{<i}, X)\end{aligned}$$

4 Transducer

To tackle the unified transduction problem, we introduce an attention-based neural transducer that extends Zhang et al. (2019)'s attention-based parser. Their attention-based parser addresses semantic parsing in a two-stage process: it first employs an extended variant of pointer-generator network (See et al., 2017) to convert the input text into a list of nodes, and then uses a deep biaffine graph-based parser (Dozat and Manning, 2016) with a maximum spanning tree (MST) algorithm to create edges. In contrast, our attention-based neural transducer directly transduces the input text into a meaning representation in *one* stage via a sequence of semantic relations. A high-level model architecture of our transducer is depicted in Fig. 2: an *encoder* first encodes the input text into hidden states; and then conditioned on the hidden states, at each decoding time step, a *decoder* takes the previous semantic relation as input, and outputs a new semantic relation, which includes a target node, a relation type, and a source node.

Specifically, there a significant difference between Zhang et al. (2019) and our model: Zhang et al. (2019) first predicts nodes, and then edges. These two stages are done *separately* (except that a shared encoder is used). At the node prediction stage, their model has no knowledge of edges, and therefore node prediction is performed purely based previous nodes. At the edge prediction stage, their model predicts the head of each node in parallel. Head prediction of one node has no constrains or impact on another. As a result, MST algorithms have to be used to search for a valid prediction. In comparison, our model does not have two separate stages for node and edge prediction. At each decoding step, our model predicts not only a node, but also the incoming edge to the node, which includes a source and a relation type. See Fig. 2 for an example. The predicted node and incoming edge together with previous predictions form a partial semantic graph, which is used as input of the next decoding step for the next node and incoming edge prediction. Our model therefore makes predictions based on the partial semantic graph, which helps prune the output space for both nodes and edges. Since at each decoding step, we assume the incoming edge is always from a preceding node (see Section 4.3 for the details), the predicted semantic graph is guaranteed to be a valid arborescence, and a MST algorithm is no longer needed.

4.1 Encoder

At the encoding stage, we employ an encoder embedding module to convert the input text into vector representations, and a BiLSTM is used to encode vector representations into hidden states.

Encoder Embedding Module concatenates word-level embeddings from GloVe (Pennington et al., 2014) and BERT[2] (Devlin et al., 2018), char-level embeddings from CharCNN (Kim et al., 2016), and randomly initialized embeddings for POS tags.

For AMR, it includes extra randomly initialized embeddings for anonymization indicators that tell the encoder whether a token is an anonymized token from preprocessing.

For UCCA, it includes extra randomly initialized embeddings for NER tags, syntactic dependency labels, punctuation indicators, and shapes that are provided in the UCCA official dataset.

Multi-layer BiLSTM (Hochreiter and Schmidhuber, 1997) is defined as:

$$\mathbf{s}_t^l = \begin{bmatrix} \overrightarrow{\mathbf{s}}_t^l \\ \overleftarrow{\mathbf{s}}_t^l \end{bmatrix} = \begin{bmatrix} \overrightarrow{\mathrm{LSTM}}(\mathbf{s}_t^{l-1}, \mathbf{s}_{t-1}^l) \\ \overleftarrow{\mathrm{LSTM}}(\mathbf{s}_t^{l-1}, \mathbf{s}_{t+1}^l) \end{bmatrix}, \qquad (1)$$

where $\mathbf{s}_t^l$ is the l-th layer hidden state at time step t; $\mathbf{s}_i^t$ is the embedding module output for token x_t.

4.2 Decoder

Decoder Embedding Module at decoding time step i converts elements in the input semantic relation $\langle u_i, d_i^u, r_i, v_i, d_i^v \rangle$ into vector representations $\langle \mathbf{u}_i, \mathbf{d}_i^u, \mathbf{r}_i, \mathbf{v}_i, \mathbf{d}_i^v \rangle$:[3]

$\mathbf{u}_i$ and $\mathbf{v}_i$ are concatenations of word-level embeddings from GloVe, char-level embeddings from CharCNN, and randomly initialized embeddings for POS tags. POS tags for source and target nodes are inferred at runtime: if a node is copied from input text, the POS tag of the corresponding token is used; if it is copied from a preceding node, the POS tag of the preceding node is used; otherwise, an `UNK` tag is used.

$\mathbf{d}_i^u, \mathbf{d}_i^v$ and $\mathbf{r}_i$ are randomly initialized embeddings for source node index, target node index, and relation type.

Next, the decoder outputs a new semantic relation in a *factorized* way depicted in Fig. 2: First, a target node module takes vector representations of the previous semantic relation, and predicts a target node label as well as its index. Then, a source node module predicts a source node via *pointing* to a preceding node. Lastly, a relation type module takes the predicted source and target nodes, and predicts the relation type between them.

Target Node Module converts vector representations of the input semantic relation into a hidden state $\mathbf{z}_i$ in the following way:

$$\mathbf{z}_i = \mathrm{FFN}^{(\mathrm{relation})}([\mathbf{h}_i^l; \mathbf{c}_i; \mathbf{r}_i; \mathbf{u}_i; \mathbf{d}_i^u]) \qquad (2)$$

$$\mathbf{h}_i^l = \mathrm{LSTM}(\mathbf{h}_i^{l-1}, \mathbf{h}_{i-1}^l) \qquad (3)$$

$$\mathrm{FFN}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b} \qquad (4)$$

where an l-layer LSTM generates contextual representation $\mathbf{h}_i^l$ for *target* node v_i (for initialization, $\mathbf{h}_i^0 = [\mathbf{v}_i; \mathbf{d}_i^v]$, $\mathbf{h}_0^l = [\overleftarrow{\mathbf{s}}_1^l; \overrightarrow{\mathbf{s}}_n^l]$). A feed-forward neural network $\mathrm{FFN}^{(\mathrm{relation})}$ generates the hidden

[2] We use average pooling in the same way as Zhang et al. (2019) to get word-level embeddings from BERT.

[3] While training, the input semantic relation is from the reference sequence of relations; at test time, it is the previous decoder output semantic relation.

state $\mathbf{z}_i$ of the input semantic relation by combining contextual representation $\mathbf{h}_i^l$ for target node v_i, encoder context vector $\mathbf{c}_i$, and vector representations $\mathbf{r}_i, \mathbf{u}_i, \mathbf{d}_i^u$ for relation type r_i, source node label u_i and source node index d_i^u.

Encoder context vector $\mathbf{c}_i$ is a weighted-sum of encoder hidden states $\mathbf{s}_{1:n}^l$. The weight is attention $\mathbf{a}_i^{(\text{enc})}$ from the decoder at decoding step i to encoder hidden states:

$$\mathbf{a}_i^{(\text{enc})} = \text{softmax}\big(\text{MLP}^{(\text{enc})}([\mathbf{h}_i^l; \mathbf{s}_{1:n}^l])\big) \quad (5)$$

$$\text{MLP}(\mathbf{x}) = \text{ELU}(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (6)$$

Given the hidden state $\mathbf{z}_i$ for input semantic relation, we use an extended variant of pointer-generator network to compute the probability distribution of next target node label v_{i+1}:

$$\text{P}(v_{i+1}) = p_{\text{gen}}\mathbf{p}_i^{(\text{vocab})} \oplus p_{\text{enc}}\mathbf{a}_i^{(\text{enc})} \oplus p_{\text{dec}}\mathbf{a}_i^{(\text{dec})} \quad (7)$$

$$\mathbf{p}_i^{(\text{vocab})} = \text{softmax}\big(\text{FFN}^{(\text{vocab})}(\mathbf{z}_i)\big) \quad (8)$$

$$\mathbf{a}_i^{(\text{dec})} = \text{softmax}\big(\text{MLP}^{(\text{dec})}([\mathbf{z}_i; \mathbf{z}_{1:i-1}])\big) \quad (9)$$

$$[p_{\text{gen}}, p_{\text{enc}}, p_{\text{dec}}] = \text{softmax}\big(\text{FFN}^{(\text{switch})}(\mathbf{z}_i)\big) \quad (10)$$

$\text{P}(v_{i+1})$ is a hybrid of three parts: (1) emitting a new node label from a pre-defined vocabulary via probability distribution $\mathbf{p}_i^{(\text{vocab})}$; (2) copying a token from the encoder input text as node label via encoder-side attention $\mathbf{a}_i^{(\text{enc})}$; and (3) copying a node label from preceding target nodes via decoder-side attention $\mathbf{a}_i^{(\text{dec})}$. Scalars $p_{\text{gen}}, p_{\text{enc}}$ and p_{dec} act as a soft switch to control the production of target node label from different sources.

The next target node index d_{i+1}^v is assigned based on the following rule:

$$d_{i+1}^v = \begin{cases} d_j^v, \text{ if } v_{i+1} \text{ copies its antecedent } v_j. \\ i+1, \text{ otherwise.} \end{cases}$$

Source Node Module produces the next source node label u_{i+1} via *pointing* to a node label among preceding *target* node labels (the dotted arrows shown in Fig. 2). The probability distribution of next source node label u_{i+1} is defined as

$$\text{P}(u_{i+1}) = \text{softmax}\big(\text{BIAFFINE}(\mathbf{h}_{i+1}^{(\text{start})}, \mathbf{h}_{1:i}^{(\text{end})})\big) \quad (11)$$

where BIAFFINE is a biaffine function (Dozat and Manning, 2016). $\mathbf{h}_{i+1}^{(\text{start})}$ is the vector representation for the *start* of the pointer. $\mathbf{h}_{1:i}^{(\text{end})}$ are vector representations for possible *ends* of the pointer. They are computed by two multi-layer perceptrons:

$$\mathbf{h}_{i+1}^{(\text{start})} = \text{MLP}^{(\text{start})}(\mathbf{h}_{i+1}^l) \quad (12)$$

$$\mathbf{h}_{1:i}^{(\text{end})} = \text{MLP}^{(\text{end})}(\mathbf{h}_{1:i}^l) \quad (13)$$

Note that $\mathbf{h}_{i+1}^l$ is the LSTM hidden state for target node v_{i+1}, generated by Equation (3) in the target node module. We reuse LSTM hidden states from the target node module such that we can train the decoder modules jointly.

Then, the next source node index d_{i+1}^u is the same as the target node the module points to.

Relation Type Module also reuses LSTM hidden states from the target node module to compute the probability distribution of next relation type r_{i+1}. Assuming that the source node module points to target node label v_j as the next source node label, The next relation type probability distribution is computed by:

$$\text{P}(r_{i+1}) = \text{softmax}\big(\text{BILINEAR}(\mathbf{h}_{i+1}^{(\text{rel-src})}, \mathbf{h}_{i+1}^{(\text{rel-tgt})})\big) \quad (14)$$

$$\mathbf{h}_{i+1}^{(\text{rel-src})} = \text{MLP}^{(\text{rel-src})}(\mathbf{h}_j^l) \quad (15)$$

$$\mathbf{h}_{i+1}^{(\text{rel-tgt})} = \text{MLP}^{(\text{rel-tgt})}(\mathbf{h}_{i+1}^l) \quad (16)$$

4.3 Training

To ensure that at each decoding step, the source node can be found in the preceding nodes, we create the reference sequence of semantic relations by running a *pre-order* traversal over the reference arborescence. The pre-order traversal only determines the order between a node and its children. As for the order of its children, we sort them in alphanumerical order in the case of AMR, following Zhang et al. (2019). In the case of SDP, we sort the children based on their order in the input text. In the case of UCCA, we sort the children based on their UCCA node ID.

Given a training pair $\langle X, Y \rangle$, the optimization objective is to maximize the decomposed conditional log likelihood $\sum_i \log\big(\text{P}(y_i \mid y_{<i}, X)\big)$, which is approximated by:

$$\sum_i \log\big(\text{P}(u_i)\big) + \log\big(\text{P}(r_i)\big) + \log\big(\text{P}(v_i)\big) \quad (17)$$

We also employ label smoothing (Szegedy et al., 2016) to prevent overfitting, and include a coverage loss (See et al., 2017) to penalize repetitive nodes: $\text{covloss}_i = \sum_t \min(\mathbf{a}_i^{(\text{enc})}[t], \mathbf{cov}_i[t])$, where $\mathbf{cov}^i = \sum_{j=0}^{i-1} \mathbf{a}_j^{(\text{enc})}$.

4.4 Prediction

Our transducer at each decoding time step looks for the source node from the preceding nodes, which ensures that the output of a greedy search is already a valid arborescence $\hat{Y}$:

$$\mathrm{P}(\hat{Y} \mid X) = \prod_i \max_{u_i} \mathrm{P}(u_i) \max_{r_i} \mathrm{P}(r_i) \max_{v_i} \mathrm{P}(v_i)$$

Therefore, a MST algorithm such as the Chu-Liu-Edmonds algorithm at $\mathcal{O}(EV)$ used in Zhang et al. (2019) is no longer needed,[4] and the decoding speed of our transducer is $\mathcal{O}(V)$. Moreover, since our transducer builds the meaning representation via a sequence of semantic relations, we implement a beam search over relation in Algo. 1. Compared to the beam search of Zhang et al. (2019) that only returns top-k nodes, our beam search finds the top-k relation scores, which includes source nodes, relation types and target nodes.

Algorithm 1: Beam Search over Semantic Relations.

```
Input  : The input text $X$.
Output: A sequence of relations $Y = \{y_1, ...y_m\}$.
// Initialization.
$i$, score $\leftarrow 0, 0$;
$Y$, finished $\leftarrow \{\}, \{\}$;
beam$\leftarrow \{\{Y, \text{score}\}\}$;
// Encoding.
encode($X$);
// Decoding.
for $i \leftarrow 1$ to MaxLength do
    new_beam $\leftarrow \{\}$;
    $\{Y, \text{score}\}$ = beam.pop();
    for $v_i$ in topK(P($v_i$)) do
        if $v_i$ = EOS then
            finished.push($\{Y, \text{score}\}$);
        else
            for $u_i \leftarrow v_0$ to $v_{i-1}$ do
                for $r_i$ in RelationTypeSet do
                    $Y \leftarrow Y \cup \{\langle u_i, r_i, v_i \rangle\}$;
                    score $\leftarrow$ score + $\log\left(\mathrm{P}(u_i)\right)$ +
                      $\log\left(\mathrm{P}(r_i)\right) + \log\left(\mathrm{P}(v_i)\right)$;
                    new_beam.push($\{$Y, score$\}$);
                end
            end
        end
    end
    beam $\leftarrow$ new_beam.topK();
end
// Finishing.
while beam.not_empty() do
    $\{Y, \text{score}\} \leftarrow$ beam.pop();
    finished.push($\{Y, \text{score}\}$);
end
$\{Y, \text{score}\} \leftarrow$ finished.topK(k=1);
return $Y$;
```

[4] E denotes the number of edges. V the number of nodes.

5 Data Pre- and Post-processing

AMR Pre- and post-processing steps are similar to those of Zhang et al. (2019): in preprocessing, we anonymize subgraphs of entities, remove senses, and convert resultant AMR graphs into the unified format; in post-processing, we assign the most frequent sense for nodes, restore Wikipedia links using the DBpedia Spotlight API (Daiber et al., 2013), add polarity attributes based on rules observed from training data, and recover the original AMR format from the unified format.

DM No pre- or post-processing is done to DM except converting them into the unified format, and recovering them from predictions.

UCCA During training, multi-sentence input text and its corresponding DAG are split into single-sentence training pairs based on rules observed from training data. At test time, we split multi-sentence input text, and join the predicted graphs into one. We also convert the original format to the unified format in preprocessing, and recover the original DAG format in post-processing.

Hidden Size		
Glove		300
BERT		1024
POS / NER / Dep / Shapes		100
Anonymization / Node index		50
CharCNN kernel size		3
CharCNN channel size		100
Encoder		2@512
Decoder		2@1024
Biaffine input size		256
Bilinear input size	AMR	128
	DM	256
	UCCA	128
Optimizer		
Type		ADAM
Learning rate		0.001
Maximum gradient norm		5.0
Coverage loss weight λ		1.0
Label smoothing ϵ		0.1
Beam size		5
Batch size		64
Dropout rate	AMR	0.33
	DM	0.2
	UCCA	0.33
Vocabulary		
Encoder-side vocab size	AMR 1.0	9200
	AMR 2.0	18000
	DM	11000
	UCCA	10000
Decoder-side vocab size	AMR 1.0	7300
	AMR 2.0	12200
	DM	11000
	UCCA	10000

Table 1: Hyperparameter settings

6 Experiments

6.1 Data and Setup

We evaluate our approach on three separate broad-coverage semantic parsing tasks: (1) AMR 2.0 (LDC2017T10) and 1.0 (LDC2014T12); (2) the English DM dataset from SemEval 2015 Task 18 (LDC2016T10); (3) the UCCA English Wikipedia Corpus v1.2 (Abend and Rappoport, 2013). The train/dev/test split follows the official setup. Our model is trained on two GeForce GTX TITAN X GPUs with early stop based on the dev set. We fix BERT parameters similar to Zhang et al. (2019) due to the limited GPU memory. Hyperparameter setting for each task is provided in Table 1.

6.2 Results

Data	Parser	F1(%)
AMR 2.0	Cai and Lam (2019)	73.2
	Lyu and Titov (2018)	74.4±0.2
	Lindemann et al. (2019)	75.3±0.1
	Naseem et al. (2019)	75.5
	Zhang et al. (2019)	76.3±0.1
	- w/o beam search	75.3±0.1
	Ours	**77.0**±0.1
	- w/o beam search	76.4±0.1
AMR 1.0	Flanigan et al. (2016)	66.0
	Pust et al. (2015)	67.1
	Wang and Xue (2017)	68.1
	Guo and Lu (2018)	68.3±0.4
	Zhang et al. (2019)	70.2±0.1
	- w/o beam search	69.2±0.1
	Ours	**71.3**±0.1
	- w/o beam search	70.4±0.1

Table 2: SMATCH F1 on AMR 2.0 and 1.0 test sets. Standard deviation is computed over 3 runs.

AMR Table 2 compares our neural transducer to the previous best results (SMATCH F1, Cai and Knight, 2013) on AMR test sets. The transducer improves the state of the art on AMR 2.0 by 0.7% F1. On AMR 1.0 where training data is much smaller than AMR 2.0, it shows a larger improvement (1.1% F1) over the state of the art.

In Table 2, we also conduct ablation study on beam search to investigate contributions from the model architecture itself and the beam search algorithm. The transducer model without beam search is already better than the previous best parser that is equipped with beam search. When compared with the previous best parser without beam search, our model still has around 1.0% F1 improvement.

Metric	L'18	N'19	Z'19	Ours
SMATCH	74	75	76	**77**
Unlabeled	77	**80**	79	**80**
No WSD	76	76	77	**78**
Reentrancies	52	56	60	**61**
Concepts	**86**	**86**	85	**86**
Named Ent.	**86**	83	78	79
Wikification	76	80	**86**	**86**
Negation	58	67	75	**77**
SRL	70	**72**	70	71

Table 3: Fine-grained F1 scores on the AMR 2.0 test set. L'18 is Lyu and Titov (2018); N'19 is Naseem et al. (2019); Z'19 is Zhang et al. (2019).

Table 3 summarizes the parser performance on each subtask using Damonte et al. (2017) evaluation tool. Our transducer outperforms Zhang et al. (2019) on all subtasks, but is still not close to Lyu and Titov (2018) on named entities due to the different preprocessing methods for anonymization.

Parser	ID	OOD
Du et al. (2015)	89.1	81.8
Almeida and Martins (2015)$^{\text{(open)}}$	89.4	83.8
Wang et al. (2018)	90.3	84.9
Peng et al. (2017a): BASIC	89.4	84.5
Peng et al. (2017a): FREDA3	90.4	85.3
Peng et al. (2018)	91.2	86.6
Dozat and Manning (2018)	**93.7**	**88.9**
Ours	92.2	87.1

Table 4: Labeled F1 (%) scores on the English DM in-domain (WSJ) and out-of-domain (Brown corpus) test sets. $^{\text{(open)}}$ denotes results from the open track.

DM Table 4 compares our neural transducer to the state of the art (labeled F1) on the English DM in-domain (ID) and out-of-domain (OOD) data. Except Dozat and Manning (2018), our transducer outperforms all other baselines, including FREDA3 of Peng et al. (2017a) and Peng et al. (2018), which leverage multi-task learning from different datasets. The best parser (Dozat and Manning, 2018) is specifically designed for bi-lexical dependencies, and is not directly applicable to other semantic parsing tasks such as AMR and UCCA. In contrast, our transducer is more general, and is competitive to the best SDP parser.

Parser	F1 (%)
Hershcovich et al. (2017)	71.1
Hershcovich et al. (2018): single	71.2
Hershcovich et al. (2018): MTL	74.3
Ours	**76.6**±0.1
- w/o non-terminal node labels	75.7±0.1

Table 5: Labeled F1 (%) scores for all edges including primary edges and remote edges. Standard deviation is computed over 3 runs.

UCCA Table 5 compares our results to the previous best published results (labeled F1 for all edges) on the English Wiki test set. Hershcovich et al. (2018) explore multi-task learning (MTL) to improve UCCA parsing, using AMR, DM and UD parsing as auxiliaries. While improvement is achieved UCCA parsing, their MTL model shows poor results on the auxiliary tasks: 64.7% unlabeled F1 on AMR, 27.2% unlabeled F1 on DM, and 4.9% UAS on UD. In comparison, our transducer improves the state of the art on AMR, and shows competitive results on DM. At the same time, it also outperforms the best published UCCA results by 2.3% F1. When converting UCCA DAGs to the unified format, we adopt a simple rule (Section 3.1) to add node labels to non-terminals. Table 5 shows that these node labels do improve the parsing performance from 75.7% to 76.6%.

6.3 Analysis

Validity Graph-based parsers like Dozat and Manning (2018); Zhang et al. (2019) make independent decisions on edge types. As a result, the same outgoing edge type can appear multiple times to a node. For instance, a node can have more than one `ARG1` outgoing edge. Although F1 scores can be computed for graphs with such kind of nodes, these graphs are in fact invalid mean representations. Our neural transducer incrementally builds meaning representations: at each decoding step, it takes a semantic relation as input, and has memory of preceding edge type information, which implicitly places constraints on edge type prediction. We compute the number of invalid graphs predicted by the parser of Zhang et al. (2019) and our neural transducer on the AMR 2.0 test set, and find that our neural transducer reduces the number of invalid graphs by 8%.

Speed Besides the improvement on parsing accuracy, we also significantly speed up parsing. Table 6 compares the parsing speed of our transducer and Zhang et al. (2019) on the AMR 2.0 test set, under the same environment setup. Without relying on MST algorithms to produce a valid arborescence, our transducer is able to parse at 1.7x speed.

	Speed (tokens/sec)
Zhang et al. (2019)	617
Ours	**1076**

Table 6: Parsing speed on the AMR 2.0 test set.

7 Conclusion

We cast three broad-coverage semantic parsing tasks into a unified transduction framework, and propose a neural transducer to tackle the problem. Given the input text, the transducer incrementally builds a meaning representation via a sequence of semantic relations. Experiments conducted on three tasks show that our approach improves the state of the art in both AMR and UCCA, and is competitive to the best parser in SDP.

This work can be viewed as a starting point for cross-framework semantic parsing. Also, compared with transition-based parsers (e.g. Damonte et al., 2017) and graph-based parsers (e.g. Dozat and Manning, 2018), our transductive framework does not require a pre-trained aligner, and it is capable of building a meaning representation that is less anchored to the input text. These advantages make it well suited to semantic parsing in cross-lingual settings (Zhang et al., 2018). In the future, we hope to explore its potential in cross-framework and cross-lingual semantic parsing.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was supported in part by the JHU Human Language Technology Center of Excellence, and DARPA LORELEI and AIDA. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238. Association for Computational Linguistics.

Mariana S. C. Almeida and André F. T. Martins. 2015. Lisbon: Evaluating TurboSemanticParser on multiple languages and out-of-domain data. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 970–973, Denver, Colorado. Association for Computational Linguistics.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710. Association for Computational Linguistics.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using stack-LSTMs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1275, Copenhagen, Denmark. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Guntis Barzdins and Didzis Gosko. 2016. RIGA at SemEval-2016 task 8: Impact of Smatch extensions and character-level neural translation on AMR parsing accuracy. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1143–1147, San Diego, California. Association for Computational Linguistics.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, pages 277–286. College Publications.

Deng Cai and Wai Lam. 2019. Core Semantic First: A Top-down Approach for AMR Parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*, Hong Kong, China. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752. Association for Computational Linguistics.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1052–1062, Sofia, Bulgaria. Association for Computational Linguistics.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.

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. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490. Association for Computational Linguistics.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015. Peking: Building semantic dependency graphs with a hybrid parser. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 927–931, Denver, Colorado. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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

Dan Flickinger, Valia Kordoni, and Zhang Yi. 2012. DeepBank: A Dynamically Annotated Treebank of the Wall Street. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*, pages 85–86, Lisbon, Portugal.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Ucl+sheffield at semeval-2016 task 8: Imitation learning for amr parsing with an alphabound. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1167–1172. Association for Computational Linguistics.

Zhijiang Guo and Wei Lu. 2018. Better transition-based amr parsing with a refined search space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722. Association for Computational Linguistics.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English dependency treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3153–3160, Istanbul, Turkey. European Language Resources Association (ELRA).

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1127–1138, Vancouver, Canada. Association for Computational Linguistics.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 373–385, Melbourne, Australia. Association for Computational Linguistics.

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

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1077–1086, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Dordrecht: Kluwer Academic Publishers.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2741–2749. AAAI Press.

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

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585, Florence, Italy. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Jonathan May. 2016. SemEval-2016 task 8: Meaning representation parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1063–1073, San Diego, California. Association for Computational Linguistics.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 536–545, Vancouver, Canada. Association for Computational Linguistics.

Yusuke Miyao and Jun'ichi Tsujii. 2004. Deep linguistic analysis for the accurate identification of predicate-argument relations. In *Proceedings of Coling 2004*, pages 1392–1398, Geneva, Switzerland. COLING.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4586–4592, Florence, Italy. Association for Computational Linguistics.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, IncrementParsing '04, pages 50–57, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72, Dublin, Ireland. Association for Computational Linguistics.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based MRS banking. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

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

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. 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 1492–1502, New Orle/ans, Louisiana. Association for Computational Linguistics.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 32–41, Beijing, China. Association for Computational Linguistics.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1143–1154. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

M. Steedman. 1996. *Surface Structure and Interpretation*. Linguistic inquiry monographs. MIT Press.

M. Steedman. 2001. *The Syntactic Process*. A Bradford book. MIT Press.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. Camr at semeval-2016 task 8: An extended transition-based amr parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178. Association for Computational Linguistics.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of amr parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268. Association for Computational Linguistics.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375. Association for Computational Linguistics.

Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. 2018. A neural transition-based approach for semantic dependency graph parsing. In *AAAI Conference on Artificial Intelligence*.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2018. Cross-lingual decompositional semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1664–1675, Brussels, Belgium. Association for Computational Linguistics.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. Amr parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689. Association for Computational Linguistics.