UC Davis at SemEval-2019 Task 1: DAG Semantic Parsing with Attention-based Decoder

Dian Yu
University of California, Davis
dianyu@ucdavis.edu

Kenji Sagae
University of California, Davis
sagae@ucdavis.edu

Abstract

We present a simple and accurate model for semantic parsing with UCCA as our submission for SemEval 2019 Task 1. We propose an encoder-decoder model that maps strings to directed acyclic graphs. Unlike many transition-based approaches, our approach does not use a state representation, and unlike graph-based parsers, it does not score graphs directly. Instead, we encode input sentences with a bidirectional-LSTM, and decode with self-attention to build a graph structure. Results show that our parser is simple and effective for semantic parsing with reentrancy and discontinuous structures.

1 Introduction

Semantic parsing aims to capture structural relationships between input strings and graph representations of sentence meaning, going beyond concerns of surface word order, phrases and relationships. The focus on meaning rather than surface relations often requires the use of reentrant nodes and discontinuous structures. Universal Conceptual Cognitive Annotation (UCCA) (Abend and Rappoport, 2013) is designed to support semantic parsing with mappings between sentences and their corresponding meanings in a framework intended to be applicable across languages.

SemEval 2019 Task 1 (Hershcovich et al., 2018b, 2019) focuses on semantic parsing of texts into graphs consisting of terminal nodes that represent words, non-terminal nodes that represent internal structure, and labeled edges representing relationships between nodes (e.g. *participant*, *center*, *linker*, *adverbial*, *elaborator*), according to the UCCA scheme. Annotated datasets are provided, and participants are evaluated in four settings: English with domain-specific data, English with out-of-domain data, German with domain-specific data, and French with only development and test data, but no training data. Additionally, there are open and closed tracks, where the use of additional resources is and is not allowed, respectively. Our entry in the task is limited to the closed track and the first setting, domain-specific English using the Wiki corpus, where the relatively small dataset (4113 sentences for training, 514 for development, and 515 for testing) consists of annotated sentences from English Wikipedia.

Our model follows the encoder-decoder architecture commonly used in state-of-the-art neural parsing models (Kitaev and Klein, 2018; Kiperwasser and Goldberg, 2016b; Cross and Huang, 2016; Chen and Manning, 2014). However, we propose a very simple decoder architecture that relies only on a recursive attention mechanism of the encoded latent representation. In other words, the decoder does not require state encoding and model-optimal inference whatsoever. Our novel model achieved a macro-averaged F1-score of 0.753 in labeled primary edges and 0.864 in unlabeled primary edge prediction on the test set. The results confirm the suitability of our proposed model to the semantic parsing task.

2 Related work

Leveraging parallels between UCCA and known approaches for syntactic parsing, Hershcovich et al. (2017) proposed TUPA, a customized transition-based parser with dense feature representation. Based on this model, Hershcovich et al. (2018a) used multitask learning effectively by training a UCCA model along with similar parsing tasks where more training data is available, such as Abstract Meaning Representation (AMR) (Banarescu et al., 2013) and Universal Dependencies (UD) (Nivre et al., 2016). Due to

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 119–124
Minneapolis, Minnesota, USA, June 6–7, 2019.

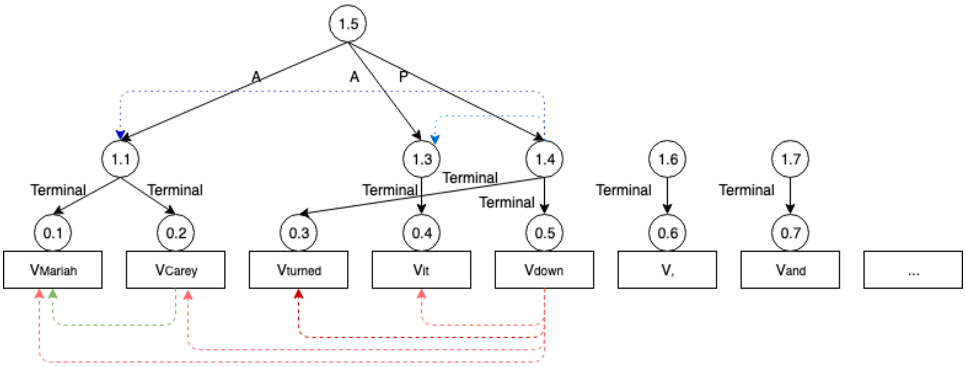

Figure 1: Illustration of the decoder for the beginning of a sentence, “Mariah Carey turned it down, and ...”. Each v_i represents the context embedding for each word i from the BiLSTM encoder. Words on edges represent category labels between nodes, where A is participant and P is process. Circles represent nodes in the graph, each with a pair in indices. Circles with 0 as the first index are terminal nodes, and circles with 1 as the first index are non-terminal nodes. (1). Dashed green lines represent the attention mechanism for the word $Carey$, which forms a continuous proper noun “Mariah Carey”. (2). Dashed red lines represent the attention mechanism for the word $down$, which forms a discontinuous unit “turned ... down”. (3). Dotted blue lines represent the attention mechanism for $node_{1.4}$. The darker the color, the higher the attention score.

the requirements of reentrancy, discontinuity, and non-terminals, other powerful parsers were shown to be less suitable for parsing with UCCA (Hershcovich et al., 2017).

3 Parsing Model

BiLSTM models are capable of providing feature representations with sequential data, and attention mechanisms (Vaswani et al., 2017) have been applied successfully to parsing tasks (Kitaev and Klein, 2018). Inspired by their success, our model uses a BiLSTM encoder and a self-attention decoder. The encoder represents each node (terminal and non-terminal) in the DAG without the need to encode features and the current parser state. The proposed decoder takes the encoded representation as the configuration and uses attention mechanism. Without any additional feature extraction, it serves a similar role as an oracle and a transition-system in transition-based parsers. We jointly train a label prediction model and a discontinuity prediction model. We predict remote edges with a different encoder. An example of the parsing model can be seen in Figure 1.

3.1 Terminal Nodes

To mitigate sparsity due to the small amount of training data available, we concatenate part-of-speech tags embeddings to word embeddings in terminal nodes. In addition, because the connections between terminal nodes and non-terminal nodes often require identification of named entities, we also added entity type and case information as additional knowledge. Given a sentence $\mathbf{x} = x_1, ..., x_n$, the vector for each input token is thus represented as $u_i = emb(x_i) \circ emb(pos_i) \circ emb(entity\_type_i) \circ emb(case_i)$, where $case_i$ is 1 if the first character of the word is capitalized and 0 otherwise. We use pretrained word embeddings from fastText[1] for $emb(x_i)$. POS tags and entity types are predicted using external models[2] and are provided in the training corpus. Each word representation from the encoder is $v_i = BiLSTM(u_i)$. We assign these contextual word embeddings as vectors to terminal nodes.

3.2 Non-terminal Nodes

For non-terminal nodes with only one terminal node as the child, the representation is the same as its corresponding terminal node, i.e. a contextual word embedding from the BiLSTM encoder. For other non-terminal nodes that have more than one terminal children or non-terminal children (i.e. represent more than one word in the text), we use a span representation. Following Cross and Huang (2016), we represent the span between the words x_i, x_j as $v_{i,j} = (f_j - f_i) \circ (b_i - b_j)$ where $f_0, ..., f_n$ and $b_0, ..., b_n$ are the output of the forward and backward directions in the BiLSTM, respectively. However, the linear subtractions from a nonlinear recurrent neural network (RNN) as a span approximation is not intuitive. Instead, we experimented

[1] https://fasttext.cc/
[2] https://spacy.io/

with an additional BiLSTM on the target span $x_i, x_{i+1}, ..., x_j$, similar to the recursive tree representations in (Socher et al., 2013; Kiperwasser and Goldberg, 2016a) but replaced the feed-forward network with an LSTM. In our experiments with the small dataset in the closed track of the English domain-specific track, this method did not result in improved performance.

3.3 Attention Mechanism For Decoding

Our basic decoding model is inspired by the global attention mechanism used in machine translation. The attention averages the encoded state in each time step in the sequence with trainable weights (Luong et al., 2015). We set a maximum sequence length and calculate the attention weights (in probability) for the left boundary index of the span given the node representation $v_{i,j}$ $(i \leq j)$:

$$h_{span} = MLP(v_{i,j}) \quad (1)$$

$$p_{left\_boundary} = softmax(h_{span}) \quad (2)$$

where MLP is a multilayer perceptron and h_{span} is of size (1, max_sequence_length). We choose $\arg\max_i p_{left\_boundary}$ as the index of the left boundary of the predicted span. Let j_l denote the index of the left most child of the node j (for example, in Figure 1, j_l for $node_{1.5}$ is 1 and j_l for $node_{1.6}$ is 6)[3]. If $i \geq j_l$, then the node attends to itself to indicate that a span cannot be created yet (as is the case for $node_{1.6}$ in Figure 1). Otherwise, there is a span that forms a semantic unit and we need to create a parent node. For example, $i = 1$ for the $node_{1.4}$, so we create a new $node_{1.5}$ which connects the nodes within the span [1: 5], i.e. $node_{1.1}$, $node_{1.3}$, and $node_{1.4}$.

We do this recursively to attend to a previous index until the node attends to itself. Then we repeat the procedure on the next word in the sequence. The illustration is shown in Figure 1 with dotted blue lines. The algorithm is presented in Algorithm 1 below. $primary\_parent$ indicates the parent node to which the current node is not a remote child (in the DAG setting, a child node may have multiple parents). We set the maximum number of recurrence to be 7 to prevent excessive node creation during inference.

Despite its simplicity, there are two limitations to this method. One is the restriction of the maximum sequence length. The other is the distinction

[3] For simplicity, word indices start at 1 in the Figure.

Algorithm 1 Index-attention decoder

1: **for** `recur_num = 1 to max_recur` **do**
2:     **if** $i \geq j_l$ **then**
3:         `break`
4:     **end if**
5:     $h_{span} = MLP(v_{i,j})$
6:     $i_{attn} = \underset{\mathtt{i}}{\mathtt{argmax}}\ softmax(h_{span})$
7:     $i = primary\_parent(v_{i_{attn}})_l$
8: **end for**

between the indices and the actual words in each sentence. The model may cheat during training to attend to specific indices regardless of the actual words in these indices.

Motivated by the success of biaffine attention(Dozat and Manning, 2016) and self-attention models (Vaswani et al., 2017), we replace the index attention decoder with a multiplication model where we can leverage fast optimized matrix multiplication. Similar to the left most child, let j_r denote the index of the right most child of $node_j$. $v_o = v[1 : j_r]$ where v is the output from the encoder of size (sequence_length, batch_size, hidden_size). The scoring function is defined as:

$$h_i = ReLU(W \times v_i + b) \quad (3)$$

$$h_o = ReLU(W \times v_o + b) \quad (4)$$

$$mm = matrix\_multiplication(h_i, h_o^T) \quad (5)$$

$$p_{left\_boundary} = softmax(mm) \quad (6)$$

Compared to the index attention decoder above, this decoder considers both the index and the span representation and thus is more flexible and robust to new texts. The recurrence call remains the same by replacing line 5 and 6 in Algorithm 1 with equations 3 − 6.

3.4 Label Prediction

Contextual information is important to label prediction. For instance, in the sentence "It announced Carey returned to the studio to start ... ", the phrase "Carey returned to the studio" should be labeled as a participant (A) instead of a scene (H) according to the context. Ideally the encoder will capture the information from the whole sentence so that we only need the current span to predict its label (since the span has the context information from both sides). However, as shown in previous research with RNN models, the contextual information is lost for a relatively long sentence. Therefore, similar to the label prediction problem with dependency parsers, we use a MLP to predict the label of a span $v_{i,j}$ given its context $p = primary\_parent(v_{i,j})$.

$$h = ReLU(W_l^1 \times (p \circ v_{i,j}) + b_l^1) \quad (7)$$

$$l = \underset{l}{argmax}\ softmax(W_l^2 * h + b_l^2) \quad (8)$$

We also experimented with only using span representation as seen in constituency parsing (Gaddy et al., 2018) by replacing $(p \circ v_{i,j})$ with $v_{i,j}$ in equation 7. Surprisingly, this increased the F1 score on the development set by 1.4 points. We conjecture that this is due to the limited amount of training data, which makes it more difficult to learn noisier representations.

3.5 Discontinuous Unit

After finding the left boundary of the current span unit as shown in section 3.3, we use two MLPs for binary classification to check (1) if the span forms a proper noun with which we need to combine multiple terminal nodes to one non-terminal node (as "Mariah Carey" in Figure 1) and (2) if the span forms a discontinuous unit (such as "turn ... down" in Figure 1).

$$prob_{propn} = W_p^2 \times ReLU(W_p^1 \times v_{i,j} + b_p^1) + b_p^2 \quad (9)$$

$$prob_{discont} = W_d^2 \times ReLU(W_d^1 \times v_{i,j} + b_d^1) + b_d^2 \quad (10)$$

If the node span attends to a node in the left and the model predicts a proper noun, we will create a non-terminal node and links all the terminal nodes $i, i+1, ..., j$ as its terminal children (shown as dashed green lines in Figure 1).

If the model predicts that the span is a discontinuous unit, instead of connecting all the terminal nodes as its children, the new created node only connects $node_i$ and $node_j$, and do the recurrence checks afterwards as shown in Algorithm 1 (illustrated as dashed red lines in Figure 1).

3.6 Remote Edges

We predict remote edges the same way as the matrix multiplication decoder for primary edges. We use a different BiLSTM encoder to learn representations and avoid confusion between attention to primary edges and remote edges.

	unlabeled(F1)	labeled(F1)
official	0.746	0.866
+ max_recur = 7	0.747	0.867
+ child_pred	0.760	0.87
+ $\beta_2 = 0.9$	0.762	0.87
+ bug_fix	**0.769**	**0.873**

Table 1: F1 score on primary edges evaluated on the development set

4 Training and Inference

During training, $node_i$ attends to the left most child of its primary parent ($node_p$) recursively until $node_p$ is not the left most child of $node_p$'s parent. Because a span representation contains information from both left to right and right to left, $node_i$ with the highest attention score not only contains the embedding of its terminal node, but also the span between index i and j in the text. We use cross entropy loss to jointly train for embeddings, the BiLSTM encoder, and the decoder.

For inference, we take the output of each token in the text from the BiLSTM encoder as input and create a non-terminal node for each terminal node. We create a new node when the token embedding attends to a different token outside of the current span boundary. The recurrence algorithm for each newly created non-terminal node shown in Algorithm 1 is applied.

5 Experiments

For the encoder, we use a 2-layer, 500 dimensional BiLSTM with 0.2 dropout. The word embedding size is 300 with feature embedding size of 20 each (pos tagging, entity type, and case information). We use Adam optimizer (Kingma and Ba, 2014) with β_2 set to 0.9 as suggested by Dozat and Manning (2016). Development set is used for early stopping. Because of the small dataset (4113 training sentences), the model overfits after 4 epochs.

6 Results

Table 1 provides the results on the development set and Table 2 shows the results on the test set. *official* shows results of the model we submitted to the competition with a maximum recursion number of 5 and a $\beta_2 = 0.99$. We obtained higher scores by increasing the recursion limit as in section 3.3 (+ max_revur = 7), using current span only

	primary		remote	
	unlabeled	labeled	unlabeled	labeled
baseline	0.733	0.858	**0.472**	**0.484**
official	0.73	**0.864**	-	-
final	**0.753**	**0.864**	0.447	0.447

Table 2: F1 score on primary and remote edges reported on the test set

as explained in section 3.4 (+ child_pred), changing β_2 as shown in section 5 (+ $\beta_2 = 0.9$) and fixing minor bugs (+ bug_fix) incrementally. *baseline* shows the results of the baseline model (TUPA) from Hershcovich et al. (2017). *final* shows the results of the model fine-tuned on the development set mentioned in Table 1.

Since there are normally 0 or 1 remote edges in each sentence in the training corpus, the remote edge prediction model is not as effective. Still, the model captures some remote relations. For example, in the sentence "Additionally, Carey's newly slimmed Figure began to change, as she stopped her exercise routines and gained weight", the node "gained weight" is predicted to point to "Carey" where the target annotated remote child is "she". Discontinuous unit prediction also suffers from the problem of insufficient training samples.

7 Conclusion

This paper describes the system that the UC Davis team submitted to SemEval 2019 Task 1. We propose a recursive self-attention decoder with a simple architecture. Our model is effective in UCCA semantic parsing, ranking third in the close track in-domain task with modest fine-tuning, highlighting the suitability of our approach.

Acknowledgments

This work was supported by the National Science Foundation under Grant No. 1840191. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the NSF.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *ACL (1)*, pages 228–238. The Association for Computer 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.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750. ACL.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. *CoRR*, abs/1612.06475.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. *CoRR*, abs/1804.07853.

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, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1127–1138.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018a. Multitask parsing across semantic representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 373–385.

Daniel Hershcovich, Zohar Aizenbud, Leshem Choshen, Elior Sulem, Ari Rappoport, and Omri Abend. 2019. Semeval 2019 task 1: Cross-lingual semantic parsing with UCCA. *CoRR*, abs/1903.02953.

Daniel Hershcovich, Leshem Choshen, Elior Sulem, Zohar Aizenbud, Ari Rappoport, and Omri Abend. 2018b. Semeval 2019 shared task: Cross-lingual semantic parsing with UCCA - call for participation. *CoRR*, abs/1805.12386.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Eliyahu Kiperwasser and Yoav Goldberg. 2016a. Easy-first dependency parsing with hierarchical tree lstms. *CoRR*, abs/1603.00375.

Eliyahu Kiperwasser and Yoav Goldberg. 2016b. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *CoRR*, abs/1603.04351.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. *CoRR*, abs/1805.01052.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. 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, Portorož, Slovenia, May 23-28, 2016.*

R Socher, A Perelygin, J.Y. Wu, J Chuang, C.D. Manning, A.Y. Ng, and C Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. *EMNLP*, 1631:1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.