Tree2Tree: A Reimplementation and Discussion of Tree-Based Neural Semantic Parsing

Alexander Fabbri
Yale University
alexander.fabbri@yale.edu

Jack Koch
Yale University
john.koch@yale.edu

Abstract

Semantic parsing is the process of mapping natural language to formal meaning representations. Recent work has applied neural end-to-end models to this task with a focus on structured representations in sequence-to-sequence problems through the use of tree decoders. In this paper, we focus on the reimplementation of a neural semantic parsing architecture and provide a comparison of the effects of encoding a natural language sequence as a tree on this existing method.

1 Introduction

Semantic parsing can be viewed as a a machine translation task between natural language input and formal meaning representations (Andreas et al., 2013). While sequence-to-sequence models with attention have achieved state of the art performance in machine translation tasks (Cho et al., 2014; Bahdanau et al., 2014; Luong et al., 2015), attempts have been made to incorporate more linguistic structure into these models (Eriguchi et al., 2016; Dong and Lapata, 2016; Marcheggiani and Titov, 2017; Chen et al., 2018). Specifically, Dong and Lapata (2016) approach the task of semantic parsing by using a tree decoder to hierarchically decode logical forms. Our work builds upon ideas in semantic parsing as well as the encoder-decoder neural network architecture and the incorporation of syntactic structures in that architecture, and we base our discussion on Dong and Lapata (2016).

In addition to using a tree decoder, recent work has attempted to encode source text using a tree structure (Tai et al., 2015; Eriguchi et al., 2016; Chen et al., 2018). We aim to combine the tree encoder from these models with the tree decoder from Dong and Lapata (2016). Chen et al. (2018) attempt to combine these two features for the task of program translation on synthetic data.

Our main contributions are the reimplementation of a neural semantic parsing architecture in PyTorch and a comparison of this architecture with one that utilizes a more structured encoder to better model the hierarchical nature of natural language. We were motivated to reimplement this architecture from Dong and Lapata (2016) because the code from their paper is in Lua/Torch, which is no longer being developed. Therefore, we believe that a reimplementation of this project, written in a major deep learning framework, will benefit the community more widely. We describe the difficulties of this reimplementation and aim to provide a fair comparison of this model with previous papers.

2 Related Work

2.1 Supervised and Semi-Supervised Semantic Parsing

A semantic parser can be learned in a supervised or semi-supervised manner in which the natural language sentence is pair with either a logical form or its denotation. In supervised semantic parsing, a natural language statement is paired with its corresponding logical meaning representation (Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010, 2011; Wong and Mooney, 2006). Due to the lack of large supervised datasets, semi-supervised semantic parsing is often performed on denotations from question-answer pairs, which are much cheaper to obtain than corresponding logical forms. These methods often use a two-staged approach in which the questions are mapped using a latent or intermediary form (Liang et al., 2013; Kwiatkowski et al., 2013; Reddy et al., 2014; Chen et al., 2017).

2.2 Neural Supervised Semantic Parsing

Additionally, recent work in deep learning has led to neural semantic parsing which aims to directly

translate natural language to logical forms (Dong and Lapata, 2016; Jia and Liang, 2016; Rabinovich et al., 2017; Yin and Neubig, 2017; Zhong et al., 2017; Xu:). Jia and Liang (2016) and Dong and Lapata (2016) represent two early examples of neural semantic parsing and form the bases of our comparisons. Jia and Liang (2016) introduce data recombination for semantic parsing in which they induce a high-precision synchronous context-free grammar from the training data and then sample from it to generate new training examples. This technique is thus able to inject prior knowledge into a recurrent neural network encoder-decoder architecture. They also add a copying mechanism inspired by pointer networks (Vinyals et al., 2015) to allow the network to copy from its input during decoding. Dong and Lapata 2016 use a similar neural network architecture, but propose a tree-based decoder to deal with the nested nature of logical forms.

2.3 Tree Models

The Recursive Neural Tensor Network was introduced to perform compositional sentiment analysis over fully-labeled parse trees (Socher et al., 2013). Expanding upon this idea, Child-Sum Tree-LSTMs and N-ary Tree-LSTMs are a form of Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) which can encode constituency and dependency parsings for the tasks of semantic relatedness and sentiment classifications (Tai et al., 2015).

Tree-based models have recently been applied to neural machine translation in the form of an attention mechanism over syntactic constituents in the encoder (Eriguchi et al., 2016). Chen et al. 2017 build upon this by introducing a bidirectional tree encoder that traverses the tree in both a bottom-up and bottom-down fashion. Additionally, they introduce a tree-coverage model that helps the attention mechanism depend on the syntax of the input. Chen et al. (2018) introduce a Tree2Tree model but apply it to the task of program translation based on synthetic data.

3 Methods

3.1 Sequence-to-Sequence Model

Our sequence-to-sequence RNN model is based on on the attention-based sequence-to-sequence model presented by Dong and Lapata (2016).

Given input $q = x_1 \cdots x_n$ and output $a = y_1 \cdots y_m$, the first n time steps belong to the encoder and the following m steps belong to the decoder. The encoder uses a word embedding function $\phi^{(\text{in})}$ each word x_i to a fixed-dimensional vector, as in Jia and Liang (2016), which are then fed as input into an LSTM network. The LSTM network has an initial hidden state h_0^F and generates a sequence of hidden states $h_1^F, ..., h_n^F$ by applying the recurrence function

$$h_i^F = \text{LSTM}(\phi^{(\text{in})}(x_i), h_{i-1}^F) \quad (1)$$

Once the tokens of the input sequence $x_1 \cdots x_n$ have been encoded into vectors, they are used to initialized the hidden states of the decoder. Then, the sequence is decoded by applying the recurrence function and the t-th output token is predicted as

$$p(y_t|y_{<t}, q) = (\text{sm}(\phi^{(\text{out})}(h_t^F))^T \mathbf{e}(y_t) \quad (2)$$

where $\phi^{(\text{out})}$ is a word embedding function, $\mathbf{e}(y_t)$ is a one-hot vector for estimating y_t's probability from the predicted distribution, and sm is the softmax function. We augment every sequence with special tokens <S> and <E> denoting the beginning and end of every sentence, respectively. During evaluation, prediction terminates when <S> is predicted, but during training, we use teacher-forcing learning. Teacher-forcing learning is when the corresponding gold-labeled token in the correct output sequence is fed into the RNN as the prediction of the RNN for the previous time step. Prediction terminates once we've iterated over every token in the target output.

3.2 Tree-based Decoders

Dong and Lapata (2016) presented a sequence-to-tree model for semantic parsing, which improved upon the results from sequence-to-sequence models by hierarchically modeling the compositional nature of meaning representations.

The tree decoder generates logical forms in a top-down manner. Following Dong and Lapata (2016), the "nonterminal" token <n> is defined to indicated subtrees. Logical forms are preprocessed into tree form by replacing tokens between pairs of brackets with nonterminals. Special tokens <s> and <(> denote the beginning of sequence and nonterminal sequence, respectively, and </s> represents the end of a sequence (including nonterminal sequences).

Essentially, the tree structure is hierarchically generated with a sequence decoder. After encoding the input query q, a sequence decoder is used to generate tokens at depth 1 of the subtree corresponding to parts of the logical form a. If the decoder predicts <n>, the corresponding nonterminal sequence is decoded by conditioning on the nonterminal's hidden vector. The process terminates when no more nonterminals are emitted. Dong and Lapata (2016) also introduce *parent-feeding* where the hidden vector of the parent nonterminal is concatenated with the inputs and fed into the RNN for each token in the subtree at the corresponding depth.

3.3 Sequence-to-Tree Model

Our sequence-to-tree model is much the same as our sequence-to-sequence model, but with the addition of the tree decoder as presented by Dong and Lapata (2016) and described in 3.2.

3.4 Tree-based Encoders

Tai et al. (2015) show that tree-based encoding gives improved results in the tasks of sentiment analysis and semantic relatedness. They propose two variants of LSTM networks, Child-Sum Tree-LSTMs and N-ary Tree-LSTMs.

3.4.1 Child-Sum Tree-LSTM

The Child-Sum Tree-LSTM is well suited for tree structures with a high branching factor or those whose children are unordered. Thus, the Child-Sum Tree-LSTM can be used to encode dependency trees. Tai et al. note that the input gate for the LSTM on a dependency tree can be interpreted as determining the semantic importance of a given word. We chose Child-Sum Tree-LSTM as our choice of tree encoder (as opposed to (Chen et al., 2018)) because dependency parses more simply capture the relationships between words in a natural language sentence than do constituency parses. We refer the reader to Tai et al. (2015) for the equations that govern Child-Sum Tree-LSTMs.

3.5 Tree-to-Tree Model

For our tree-to-tree model, we used the same tree decoder, but instead of encoding the input query as a sequence with an RNN, we generated a dependency parse tree of the natural language input using the Stanford CoreNLP parser (Manning et al., 2014), which we then feed into a Child-Sum Tree LSTM (Tai et al., 2015).

4 Experiments

4.1 Datasets

Our model was trained on the following standard datasets for semantic parsing. These are the same datasets as in Dong and Lapata (2016) except the IFTTT dataset (Quirk et al., 2015) as the preprocessed data was not included in their online code.

4.1.1 GEO

The GEO dataset (Wong and Mooney, 2006) contains natural language queries about U.S. geography and their associated Prolog queries. We use the standard split of 600 training examples and 280 test examples (Zettlemoyer and Collins, 2005).

4.1.2 JOBS

The JOBS dataset contains queries to a database of job listings and their associated Prolog-style queries. We use the split of 500 training and 140 as in Zettlemoyer and Collins (2005).

4.1.3 ATIS

The ATIS dataset contains contains natural language queries to a flights database paired with lambda-calculus queries. We use the split of 4473 training examples and 448 test used by Zettlemoyer and Collins (2007).

4.2 Settings

All models were implented in PyTorch. Our aim was to as closely match the implementations of Seq2Seq and Seq2Tree in Dong and Lapata (2016) as possible. Therefore, we used the same hyperparameters and settings as them for these models. For Tree2Tree, we started with Seq2Tree hyperparameters and then tried adjusting various settings to improve performance, including the learning rate, learning rate decay, and the number of training epochs, among others. Accuracy here is defined as sequence-level accuracy.

4.3 Results

Model	D&L 2016	Our Accuracy
Seq2Seq	84.6	82.5
-attention	72.9	70.0
Seq2Tree	87.1	86.1
-attention	76.8	74.6
Tree2Tree	-	82.1
-attention	-	78.2

Table 1: Results on the GEO dataset

Model	D&L 2016	Our Accuracy
Seq2Seq	87.1	85.7
-attention	77.9	75.0
Seq2Tree	90.0	87.9
-attention	83.6	85.7
Tree2Tree	-	80.7
-attention	-	81.4

Table 2: Results on the Jobs dataset

Model	D&L 2016	Our Accuracy
Seq2Seq	84.2	82.5
-attention	75.7	76.1
Seq2Tree	84.6	80.1
-attention	77.5	77.7
Tree2Tree	-	77.0
-attention	-	76.8

Table 3: Results on the ATIS dataset

Method	Accuracy
Seq2Seq (Jia and Liang, 2016)	77.1
without copy, with greedy search	63.6
Seq2Seq (Dong and Lapata, 2016)	84.6
Seq2Tree (Dong and Lapata, 2016)	87.1

Table 4: Others' Results on the GEO Dataset

A comparison of sequence-level accuracies Dong and Lapata (2016) and our reimplementation for the GEO, JOBS, and ATIS datasets can be found in Table 1, Table 2 and Table 3, respectively, where -attention refers to the baseline model without an attention mechanism. The accuracies obtained through our reimplementation are within a range of a couple of points of the original performance from Dong and Lapata (2016). We attribute this discrepancy to the difference in random initialization between Pytorch and Lua/Torch. Notably our vanilla Seq2Seq and Seq2Tree models are closest to the original performance on the ATIS dataset, which is the largest of the datasets and thus less susceptible to changes in random initialization.

For Tree2Tree without attention, only our implementation on the GEO dataset outperformed our Seq2Tree model. We attribute this improvement on GEO to the length of its natural language queries compared to JOBS and ATIS; GEO's longer queries allow us to effectively model them using a tree structure, which is not as effective on shorter queries. Additionally, we did not see improvement when we added an attention mechanism to the vanilla Tree2Tree model. GEO and ATIS showed minor improvements with attention, but the improvements were minimal compared to the addition of attention on other models. A potential problem is that tree attention mechanisms tend to attend repeatedly to the same region of the input (Chen et al., 2017). Chen et al. (2017) introduce a coverage mechanism to better attend to input, and we leave this for future work.

Finally, we would like to draw attention to the comparability of results between Dong and Lapata (2016) and Jia and Liang (2016). A comparison of their performances is found in 4.3. We report only GEO as a case study. While both papers implement Seq2Seq models, the performances are vastly different. Jia and Liang (2016) obtain 77.1 sequence-level accuracy, noticeably lower than the Dong and Lapata (2016). However, this model already has the addition of their novel (not present in Dong and Lapata (2016)) copy mechanism to copy source tokens during predicting, and they also perform beam search to choose the best parses which execute without error against an engine. When these functionalities are removed to match the use of greedy search by Dong and Lapata (2016), the difference in performance is even starker. Jia and Liang (2016) used bidirectional recurrent neural networks versus unidirectional, from which one would expect better performance. We attribute the large performance gap to other differences: Dong and Lapata (2016) perform extensive preprocessing to stem words and replace entities with standardized variables beyond the standardization in Jia and Liang (2016). Additionally, Dong and Lapata (2016) perform batch optimization versus the online optimization seen in Jia and Liang (2016).

5 Conclusion

In this paper, we focus on the reimplementation of the sequence-to-tree architecture from Dong and Lapata (2016) and provide a comparison of the effects of encoding a natural language sequence as a tree to this method. Additionally, our analysis of two published implementations of a sequence-to-sequence architecture for the same semantic parsing task reaffirms the difficulty of making comparisons between published results, even on such similar architectures, due to factors that are little more than footnotes in these papers, such as data preprocessing.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 47–52.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. *CoRR*, abs/1707.05436.

Xinyun Chen, Chang Liu, and Dawn Song. 2018. Tree-to-tree neural networks for program translation. *CoRR*, abs/1802.03691.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *CoRR*, abs/1601.01280.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. *CoRR*, abs/1603.06075.

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

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *CoRR*, abs/1606.03622.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1545–1556.

Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1223–1233.

Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1512–1523.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

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

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP*.

Chris Quirk, Raymond J. Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 878–888.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. 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 1139–1149.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *TACL*, 2:377–392.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 4-9, 2006, New York, New York, USA*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. *CoRR*, abs/1704.01696.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007*, pages 678–687.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.