Semantic Parsing for Task Oriented Dialog using Hierarchical Representations

Sonal Gupta **Rushin Shah** **Mrinal Mohit** **Anuj Kumar**
Facebook Conversational AI

Michael Lewis
Facebook AI Research
{sonalgupta, rushinshah, mrinalmohit, anujk, mikelewis}@fb.com

Abstract

Task oriented dialog systems typically first parse user utterances to semantic frames comprised of intents and slots. Previous work on task oriented intent and slot-filling work has been restricted to one intent per query and one slot label per token, and thus cannot model complex compositional requests. Alternative semantic parsing systems have represented queries as logical forms, but these are challenging to annotate and parse. We propose a hierarchical annotation scheme for semantic parsing that allows the representation of compositional queries, and can be efficiently and accurately parsed by standard constituency parsing models. We release a dataset of 44k annotated queries [1], and show that parsing models outperform sequence-to-sequence approaches on this dataset.

1 Introduction

Intelligent personal assistants are now ubiquitous, but modeling the semantics of complex compositional natural language queries remains challenging. Typical systems classify the *intent* of a query (e.g. `GET_DIRECTIONS`) and tag the necessary slots (e.g. `San Francisco`) (Mesnil et al., 2013; Liu and Lane, 2016). It is difficult for such representations to adequately represent nested queries such as *"Driving directions to the Eagles game"*, which is composed of `GET_DIRECTIONS` and `GET_EVENT` intents. We explore a hierarchical representation for such queries, which dramatically improves the expressive power while remaining accurate and efficient to annotate and parse (see Figure 1).

We introduce a Task Oriented Parsing (TOP) representation for intent-slot based dialog systems. This hierarchical representation is expressive enough to capture the semantics of complex nested queries, but is easier to annotate and parse than alternative representations such as logical forms or dependency graphs. We show empirically that our representation is expressive enough to model the vast majority of human-generated requests in two domains.

A key advantage of our representation is that it has a structure similar to standard constituency parses, allowing us to easily adapt algorithms developed for phrase structure parsing for inference. In particular, we use linear-time Recurrent Neural Network Grammars (RNNG) (Dyer et al., 2016) and show that the inductive bias provided by this model significantly improves the accuracy compared to strong sequence-to-sequence (seq2seq) models based on CNNs, LSTMs and Transformers.

Our contributions in this paper are:

1. A hierarchical semantic representation for task oriented dialog systems that can model compositional and nested queries.
2. A publicly available dataset of 44k requests annotated with our representation. We show that our representation has very high coverage of these requests, and that inter-annotator agreement is high.
3. We show that the representation is learnable by standard algorithms. In particular, we show that the RNNG parsing model outperforms seq2seq baselines.

2 Representation

Designing semantic annotation schemes requires trade-offs between how expressive the representation is on one hand, and how easily can it be annotated, parsed, and executed on the other. Most existing annotations for task oriented dialog systems have fallen on the extremes of non-recursive intent and slot tagging, such as in the ATIS dataset (Mes-

[1] http://fb.me/semanticparsingdialog

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2787–2792
Brussels, Belgium, October 31 - November 4, 2018.

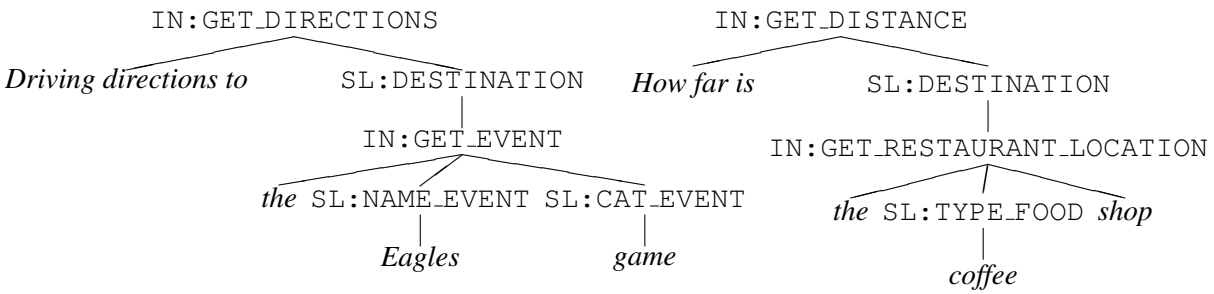

Figure 1: Example TOP annotations of utterances. Intents are prefixed with `IN:` and slots with `SL:`. In a traditional intent-slot system, the `SL:DESTINATION` could not have an intent nested inside it.

nil et al., 2013; Liu and Lane, 2016), and full logical forms (Zettlemoyer and Collins, 2012).

We introduce a hierarchical representation, similar to a constituency syntax tree, with words as terminals. Non-terminals are either *intents* or *slots*, and the root node is an intent. We allow intents to be nested inside a slot, resulting in the ability to compose requests and call multiple APIs. Using this compositional tree representation, we can enable answering compositional queries over multiple domains.

We introduce the following constraints in our representation: 1. The top level node must be an intent, 2. An intent can have tokens and/or slots as children, 3. A slot can have either tokens as children or one intent as a child.

Executing queries such as those in Figure 1 is straightforward because of the explicit tagging of the outer location slot: first we fetch 'the Eagles game' event (or the relevant coffee shop), extract the location, and pass it as the destination slot to the navigation domain intent.

Compositional queries are frequent. In our dataset of crowd-sourced utterances, we found that 30% could not be adequately represented with traditional intent-slot tagging.[2] This shows that more expressive representations are often necessary.

While our representation is capable of modeling many complex queries, some utterances are beyond its scope. For example, in *Set an alarm at 8 am for Monday and Wednesday*, *8 am* needs to be associated with both *Monday* and *Wednesday* which would require graph-structured representations. However, we found that just 0.3% of our dataset would require a more expressive representation to model adequately.[3] More expressive representations, such as dependency graphs or logical forms, would only bring marginal gains but would add significant challenges for annotation and learning.

Together, these results suggest that our tree-structured approach offers a useful compromise between traditional intent-slot tagging and logical forms, by providing very high coverage of queries while avoiding the complexities of annotating and learning more expressive representations.

In summary, our representation has the following attractive properties:

- **Expressiveness** Compared to traditional intent-slot annotations, it can express complex hierarchical queries, improving coverage of queries by 30%. We found that more general representations are required for only 0.3% of queries.
- **Easy Annotation** Annotating our representation simply requires labeling spans of a sentence, which is much more straightforward than alternatives such as creating a logical form for the sentence, or an arbitrary dependency graph. This fact allows us to quickly create a large dataset.
- **Efficient and Accurate Parsing** Since our representation closely resembles syntactic trees, we can easily re-use models from the large literature on constituency parsing.

[2] In our dataset, 35% of queries have depth >2, which means that the traditional intent-slot tagging systems would not have been able to annotate or predict these annotations. In addition, to avoid the depth-based statistic being influenced by our label set specification, we manually performed analysis of 100 samples that showed that 30% of the queries required compositional representation.

[3] Utterances that could not be annotated with the label sets and corresponding instructions were marked as 'unsupported' (9.14% of the dataset), including the ones that needed more expressive representation than a tree. Analysis of a random sample of 120 unsupported utterances shows that only four instances cannot be supported because of the tree representation constraint, estimating that there are only 0.3% queries that would require a more general (e.g graph-based) representation.

- **Execution** Our approach can be seen as a simple generalization of traditional dialog systems, meaning that existing infrastructure can easily be adapted to execute the intents.

3 Dataset

We asked crowdsourced workers to generate natural language sentences that they would ask a system that could assist in navigation and event queries. These requests were then labeled by two annotators. If these annotations weren't identical then we adjudicated with a third annotator. If all three annotators disagreed then we discarded the utterance and its annotations. 63.40% of utterances were resolved with 2 annotations and 94.09% were resolved after getting 3 annotations. We also compared percentage of utterances that were resolved after 2 annotations for depth ≤ 2 (traditional slot filling) and for depth > 2 (compositional): 68.87% vs 62.03%, noting that the agreement rate is similar.

We collected a total of 44783 annotations with 25 intents and 36 slots, randomly split into 31279 training, 4462 validation and 9042 test utterances. The dataset has utterances that are focused on navigation, events, and navigation to events.

The median (mean) depth of the trees is 2 (2.54), and the median (mean) length of the utterances is 8 (8.93) tokens. 35% of trees have depth more than 2. The dataset has 4646 utterances that contain both navigation and event intents. Figure 2 shows the distribution of instances in the full dataset over the utterance length and tree depth.

4 Models

We experiment with two types of models: standard sequence-to-sequence learning models, and a model adapted from syntactic parsing, Recurrent Neural Network Grammars (Dyer et al., 2016) (RNNG). RNNG is a top-down transition-based parser and was originally proposed for parsing syntax trees and language modeling. We trained the RNNG parser discriminatively and not generatively to reduce training time of the model. While sequence-to-sequence learning can model arbitrary sequence transduction, we hypothesize that parsing models like RNNG, which can only output well-formed trees, will give better inductive bias and flexibility for predicting compositional and cross-domains scenarios on the fly, particularly for domains with less training data available.

We briefly review the RNNG model – The parse tree is constructed using a sequence of transitions, or 'actions'. The transitions are defined as a set of SHIFT, REDUCE, and the generation of intent and slot labels. SHIFT action *consumes* an input token (that is, adds the token as a child of the right most 'open' sub-tree node) and REDUCE *closes* a sub-tree. The third set of actions is generating non-terminals – the slot and intent labels. Note that at each step, there are only a subset of valid actions. For examples, if all the input tokens have been added to the tree, the only valid action is REDUCE.

5 Experiments and Results

Baselines

We compared RNNG with implementations of sequence-to-sequence models using CNNs (Gehring et al., 2017), LSTMs (Wiseman and Rush, 2016) and Transformer networks (Vaswani et al., 2017) in fairseq[4].

Metrics

We used three metrics to evaluate the systems. **Exact match** accuracy is defined as the number of utterances whose full trees are correctly predicted. The second metric is a commonly used scoring method for syntactic parsing – **labeled bracketing** F_1 scores (Black et al., 1991) (called F_1 henceforth). We used the pre-terminals as well in the calculations. One downside of this metric is that it only considers the token spans for a given non-terminal but not the internal structure. Tree substructures are rather important for task completion. Thus, we introduce a third metric, **Tree-Labeled** (TL) F_1, which compares the sub-tree structure for a non-terminal, instead of just the token span. Exact match accuracy is the strictest metric and F_1 is the least strict metric. **Tree Validity** is the percentage of predictions which formed valid trees (via bracket matching).

Preprocessing and Hyperparameters

We used the same preprocessing of unknown words as used in (Dyer et al., 2016) and mapped numbers to a constant. We used pre-trained GloVe embeddings (Pennington et al., 2014) and tuned hyperparameters on the validation set.

- **RNNG** - We used 2-layer 164-unit LSTM with a bidirectional LSTM compositional

[4]https://github.com/pytorch/fairseq

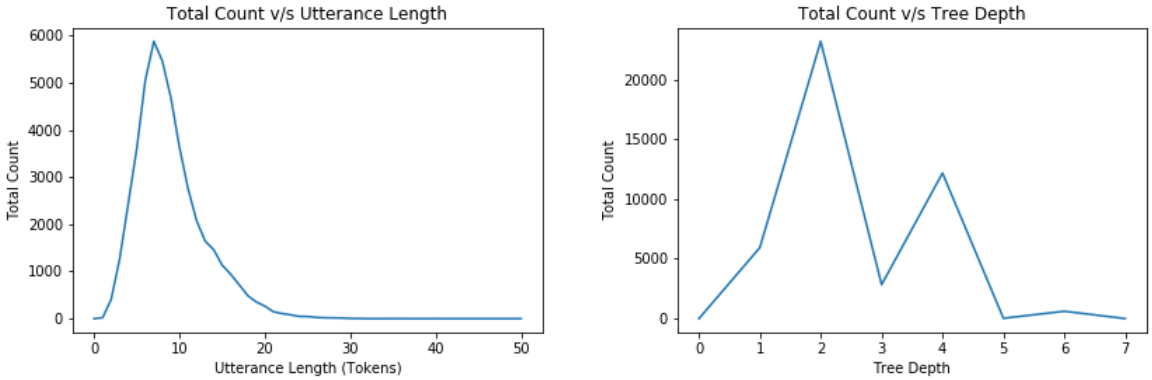

Figure 2: Count statistics of the full dataset.

Model	Exact match	F_1	Precision	Recall	TL-F_1	TL-Precision	TL-Recall	Tree Validity
RNNG	**78.51**	**90.23**	**90.62**	**89.84**	**84.27**	**84.64**	**83.91**	**100.00**
seq2seq-CNN (LOTV)	75.87	88.56	89.25	87.88	82.31	82.92	81.72	99.75
seq2seq-LSTM (LOTV)	75.31	87.69	88.35	87.03	81.15	81.72	80.58	99.94
seq2seq-Transformer	72.20	86.60	87.09	86.11	78.54	78.99	78.19	99.55

Table 1: Performance (in percentage) of RNNG and seq2seq models based on LSTMs, CNNs, and Transformer networks. The CNN and LSTM models were trained with a Limited Output Token Vocabulary (LOTV) of just a single element.

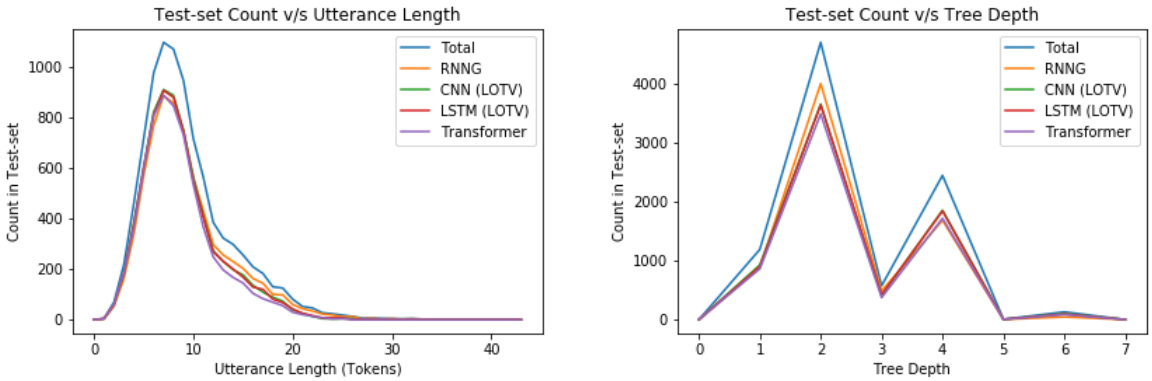

Figure 3: Exact match statistics of the different models on the testset

function, trained with a learning rate of 0.0004, weight decay of 0.00004, dropout of 0.34 with Adam optimizer for 1 epoch with 16 workers using Hogwild updates.

- **seq2seq CNN** - We used 3x3 convolutions (9 layers of 512 units, 4 units of 1024 units), followed by 1x1 convolutions (2 layers of 2048 units) with attention, trained with initial learning rate of 0.9, dropout of 0.2, gradient clipping of 0.1 with NAG optimizer for 30 epochs, inferred with beam size 5.
- **seq2seq Transformer** - We used 3 layers of 4 FFN attention heads of embedding dimension 512, trained with initial learning rate of 0.01, dropout of 0.2, gradient clipping of 5 with Adam optimizer for 50 epochs, inferred with beam size 5.
- **seq2seq LSTM** - We used 1-layer 256 unit LSTMs with attention, trained with initial learning rate of 0.7, dropout of 0.2, gradient clipping of 0.1 with NAG optimizer for 40 epochs, inferred with beam size 5.

Results

The experimental results in Table 1 and Figure 3 show that existing approaches for syntactic parsing, such as RNNG, perform well for this task, achieving perfect outputs on over 75% of queries. RNNG performs better than sequence-to-sequence

models, especially in predicting exact trees, which is important for task completion. We present results for the CNN and LSTM models with an output terminal token vocabulary of just a single element (LOTV), which performed better than the regular token vocabulary (exact match accuracy of 75.63% and 68.39%, respectively). We believe LOTV makes the models focus on learning to predict the tree structure rather than to reproduce the input tokens. But we observed that this vocabulary reduction resulted in significantly poorer performance for the Transformer model.

Below we discuss how varying the beam size during inference affects accuracy. The seq2seq results in Table 1 are accuracy for the top prediction when a beam of size 5 was run and the RNNG results are for greedy inference. For RNNG, the accuracy of top prediction did not change much when a beam of size 5 was run. We also measured how often the correct tree annotation was in the top k predictions for the models when a beam of size k was run during inference, called as *Top-k*. For RNNG, *Top-3* was 90.21 and *Top-5* was 92.48, compared to 78.51 for k=1. For seq2seq-CNN, the *Top-3* score was 88.08 and *Top-5* score was 90.21. For seq2seq-LSTM, *Top-3* was 86.55 and *Top-5* was 88.76. We note that *Top-5* is substantially higher than the accuracy of the top prediction. These top-k predictions could be used by a hypothesis ranker downstream, which can take into account agent capabilities.

We also experimented with more minimal representations of the RNNG model (Kuncoro et al., 2017). Removing the actions LSTM dropped Exact match score slightly to 78.08. Separately, removing the stack LSTM dropped it to 75.31. Removing the buffer LSTM caused a unusable decrease to 13.78.

While sequence-to-sequence models have shown strong parsing performance when trained on very large amounts of data (Vinyals et al., 2015); in our setting the inductive bias provided by the RNNG model is crucial to achieving high performance. The model has several useful biases, such as guaranteeing a well-formed output tree, and shortening the dependencies between intents and their slots.

A further advantage of RNNG is that inference has linear time complexity, whereas seq2seq models are quadratic because attention is recomputed at every time step.

6 Related Work

Many annotation schemes have previously been proposed for representing the semantics of natural language. We briefly compare our method with these.

Most work on task oriented dialog systems has focused on identifying a single user intent and then filling the relevant slots – for example, the representations used on the ATIS dataset (Mesnil et al., 2013; Liu and Lane, 2016; Zhu and Yu, 2017) and in the Dialog State Tracking Challenge (Williams et al., 2016). We showed that hierarchical representations with nested intents can improve coverage of requests in our domains.

The semantic parsing literature has focused on representing language with logical forms (Liang, 2016; Zettlemoyer and Collins, 2012; Kwiatkowski et al., 2010). Logical forms are more expressive than our representation, as they are less tightly coupled to the input query, and can be executed directly. However, logical forms are difficult to annotate and no large-scale datasets are available.

While we used a tree-structured representation, others have used arbitrary graphs, such as Abstract Meaning Representation (Banarescu et al., 2013) and Alexa Meaning Representation (Fan et al., 2017). These approaches can represent complex constructions that are beyond the scope of our approach, but with significantly challenging parsing (Artzi et al., 2015). We showed that such cases are very rare in our data.

7 Conclusions

Drawing on ideas from slot-filling and semantic parsing, we introduce a hierarchical generalization of traditional intents and slots that allows the representation of complex nested queries, leading to 30% higher coverage of user requests. We show that the representation can be annotated with high agreement. We are releasing a large dataset of annotated utterances at `http://fb.me/semanticparsingdialog`. The representation allows the use of existing constituency parsing algorithms, resulting in higher accuracy than sequence-to-sequence models.

References

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.

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.

Ezra W. Black, Steven Abney, Daniel P. Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert J. P. Ingria, Frederick Jelinek, Judith L. Klavans, Mark Y. Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proc. DARPA Speech and Natural Language Workshop, pp. 306311.*

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL.*

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, ACL.*

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. *ArXiv e-prints*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Tom Kwiatkowski, Luke 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*, pages 1223–1233. Association for Computational Linguistics.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Commun. ACM*, 59(9):68–76.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *INTERSPEECH*.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*.

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.

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

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2773–2781, Cambridge, MA, USA. MIT Press.

Jason Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.

S. Zhu and K. Yu. 2017. Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5675–5679.