Practical Semantic Parsing for Spoken Language Understanding

Marco Damonte*
University of Edinburgh
m.damonte@sms.ed.ac.uk

Rahul Goel **Tagyoung Chung**
Amazon Alexa AI
{goerahul,tagyoung}@amazon.com

Abstract

Executable semantic parsing is the task of converting natural language utterances into logical forms that can be directly used as queries to get a response. We build a transfer learning framework for executable semantic parsing. We show that the framework is effective for Question Answering (Q&A) as well as for Spoken Language Understanding (SLU). We further investigate the case where a parser on a new domain can be learned by exploiting data on other domains, either via multi-task learning between the target domain and an auxiliary domain or via pre-training on the auxiliary domain and fine-tuning on the target domain. With either flavor of transfer learning, we are able to improve performance on most domains; we experiment with public data sets such as Overnight and NLmaps as well as with commercial SLU data. We report the first parsing results on Overnight and state-of-the-art results on NLmaps. The experiments carried out on data sets that are different in nature show how executable semantic parsing can unify different areas of NLP such as Q&A and SLU.

1 Introduction

Due to recent advances in speech recognition and language understanding, conversational interfaces such as Alexa, Cortana, and Siri are becoming more common. They currently have two large uses cases. First, a user can use them to complete a specific task, such as playing music. Second, a user can use them to ask questions where the questions are answered by querying knowledge graph or database back-end. Typically, under a common interface, there exist two disparate systems that can handle each use cases. The system underlying the first use case is known as a spoken language understanding (SLU) system. Typical commercial SLU systems rely on predicting a coarse user intent and then tagging each word in the utterance to the intent's slots. This architecture is popular due to its simplicity and robustness. On the other hand, Q&A, which need systems to produce more complex structures such as trees and graphs, requires a more comprehensive understanding of human language.

One possible system that can handle such task is an executable semantic parser (Liang, 2013; Kate et al., 2005). Given a user utterance, an executable semantic parser can generate tree or graph structures that represent logical forms that can be used to query a knowledge base or database. In this work, we propose executable semantic parsing as a common framework for both uses cases by framing SLU as executable semantic parsing that unifies the two use cases. For Q&A, the input utterances are parsed into logical forms that represent the machine-readable representation of the question, while in SLU, they represent the machine-readable representation of the user intent and slots. One added advantage of using parsing for SLU is the ability to handle more complex linguistic phenomena such as coordinated intents that traditional SLU systems struggle to handle (Agarwal et al., 2018). Our parsing model is an extension of the neural transition-based parser of Cheng et al. (2017).

A major issue with semantic parsing is the availability of the annotated logical forms to train the parsers, which are expensive to obtain. A solution is to rely more on distant supervisions such as by using question–answer pairs (Clarke et al., 2010; Liang et al., 2013). Alternatively, it is possible to exploit annotated logical forms from a different domain or related data set. In this paper, we focus on the scenario where data sets for several domains exist but only very little data for a new one is available and apply transfer learning tech-

*Work conducted while interning at Amazon Alexa AI.

niques to it. A common way to implement transfer learning is by first pre-training the model on a domain on which a large data set is available and subsequently fine-tuning the model on the target domain (Thrun, 1996; Zoph et al., 2016). We also consider a multi-task learning (MTL) approach. MTL refers to machine learning models that improve generalization by training on more than one task. MTL has been used for a number of NLP problems such as tagging (Collobert and Weston, 2008), syntactic parsing (Luong et al., 2015), machine translation (Dong et al., 2015; Luong et al., 2015) and semantic parsing (Fan et al., 2017). See Caruana (1997) and Ruder (2017) for an overview of MTL.

A good Q&A data set for our domain adaptation scenario is the Overnight data set (Wang et al., 2015b), which contains sentences annotated with Lambda Dependency-Based Compositional Semantics (Lambda DCS; Liang 2013) for eight different domains. However, it includes only a few hundred sentences for each domain and its vocabularies are relatively small. We also experiment with a larger semantic parsing data set (NLmaps; Lawrence and Riezler 2016). For SLU, we work with data from a commercial conversational assistant that has a much larger vocabulary size. One common issue in parsing is how to deal with rare or unknown words, which is usually addressed by either delexicalization or by implementing a copy mechanism (Gulcehre et al., 2016). We show clear differences in the outcome of these and other techniques when applied to data sets of varying sizes. Our contributions are as follows:

- We propose a common semantic parsing framework for Q&A and SLU and demonstrate its broad applicability and effectiveness.
- We report strong parsing baselines for Overnight for which parsing scores have not been yet published and state-of-the-art results on NLmaps.
- We show that SLU greatly benefits from a copy mechanism, which is also beneficial for NLmaps but not Overnight.
- We investigate the use of transfer learning and show that it can facilitate parsing on low-resource domains.

2 Transition-based Parser

Transition-based parsers are widely used for dependency parsing (Nivre, 2008; Dyer et al., 2015) and they have been also applied to semantic parsing tasks (Wang et al., 2015a; Cheng et al., 2017).

In syntactic parsing, a transition system is usually defined as a quadruple: $T = \{S, A, I, E\}$, where S is a set of states, A is a set of actions, I is the initial state, and E is a set of end states. A state is composed of a buffer, a stack, and a set of arcs: $S = (\beta, \sigma, A)$. In the initial state, the buffer contains all the words in the input sentence while the stack and the set of subtrees are empty: $S_0 = (w_0|\dots|w_N, \emptyset, \emptyset)$. Terminal states have empty stack and buffer: $S_T = (\emptyset, \emptyset, A)$.

During parsing, the stack stores words that have been removed from the buffer but have not been fully processed yet. Actions can be performed to advance the transition system's state: they can either consume words in the buffer and move them to the stack (SHIFT) or combine words in the stack to create new arcs (LEFT-ARC and RIGHT-ARC, depending on the direction of the arc)[1]. Words in the buffer are processed left-to-right until an end state is reached, at which point the set of arcs will contain the full output tree.

The parser needs to be able to predict the next action based on its current state. Traditionally, supervised techniques are used to learn such classifiers, using a parallel corpus of sentences and their output trees. Trees can be converted to states and actions using an oracle system. For a detailed explanation of transition-based parsing, see Nivre (2003) and Nivre (2008).

2.1 Neural Transition-based Parser with Stack-LSTMs

In this paper, we consider the neural executable semantic parser of Cheng et al. (2017), which follows the transition-based parsing paradigm. Its transition system differs from traditional systems as the words are not consumed from the buffer because in executable semantic parsing, there are no strict alignments between words in the input and nodes in the tree. The neural architecture encodes the buffer using a Bi-LSTM (Graves, 2012) and the stack as a Stack-LSTM (Dyer et al., 2015), a recurrent network that allows for push and pop

[1] There are multiple different transition systems. The example we describe here is that of *arc-standard* system (Nivre, 2004) for projective dependency parsing.

operations. Additionally, the previous actions are also represented with an LSTM. The output of these networks is fed into feed-forward layers and softmax layers are used to predict the next action given the current state. The possible actions are REDUCE, which pops an item from the stack, TER, which creates a terminal node (i.e., a leaf in the tree), and NT, which creates a non-terminal node. When the next action is either TER or NT, additional softmax layers predict the output token to be generated. Since the buffer does not change while parsing, an attention mechanism is used to focus on specific words given the current state of the parser.

We extend the model of Cheng et al. (2017) by adding character-level embeddings and a copy mechanism. When using only word embeddings, out-of-vocabulary words are usually mapped to one embedding vector and do not exploit morphological features. Our model encodes words by feeding each character embedding onto an LSTM and concatenate its output to the word embedding:

$$x = \{e_w; h_c^M\}, \quad (1)$$

where e_w is the word embedding of the input word w and h_c^M is the last hidden state of the character-level LSTM over the characters of the input word $w = c_0, \ldots, c_M$.

Rare words are usually handled by either delexicalizing the output or by using a copy mechanism. Delexicalization involves substituting named entities with a specific token in an effort to reduce the number of rare and unknown words. Copy relies on the fact that when rare or unknown words must be generated, they usually appear in the same form in the input sentence and they can be therefore copied from the input itself. Our copy implementation follows the strategy of Fan et al. (2017), where the output of the generation layer is concatenated to the scores of an attention mechanism (Bahdanau et al., 2015), which express the relevance of each input word with respect to the current state of the parser. In the experiments that follow, we compare delexicalization with copy mechanism on different setups. A depiction of the full model is shown in Figure 1.

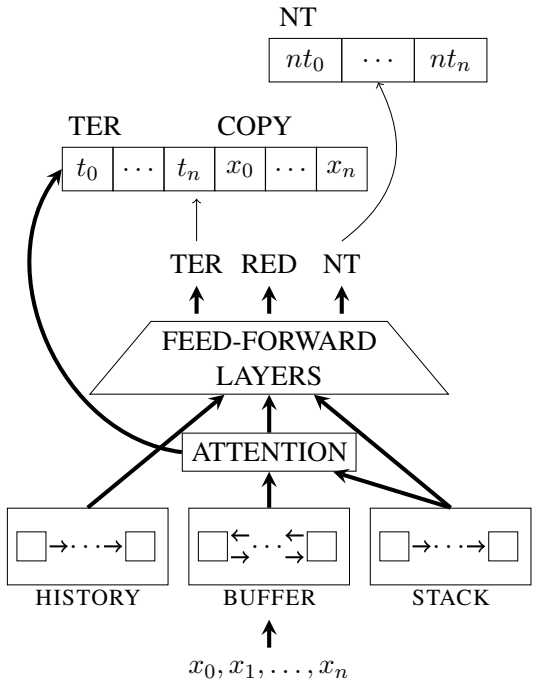

Figure 1: The full neural transition-based parsing model. Representations of stack, buffer, and previous actions are used to predict the next action. When the TER or NT actions are chosen, further layers are used to predict (or copy) the token.

3 Transfer learning

We consider the scenario where large training corpora are available for some domains and we want to bootstrap a parser for a new domain where little training data is available. We investigate the use of two transfer learning approaches: pre-training and multi-task learning.

For MTL, the different tasks share most of the architecture and only the output layers, which are responsible for predicting the output tokens, are separate for each task. When multi-tasking across domains of the same data set, we expect that most layers of the neural parser, such as the ones responsible for learning the word embeddings and the stack and buffer representation, will learn similar features and can, therefore, be shared. We implement two different MTL setups: a) when separate heads are used for both the TER classifier and the NT classifier, which is expected to be effective when transferring across tasks that do not share output vocabulary; and b) when a separate head is used only for the TER classifier, more appropriate when the non-terminals space is mostly shared.

4 Data

In order to investigate the flexibility of the executable semantic parsing framework, we evaluate models on Q&A data sets as well as on

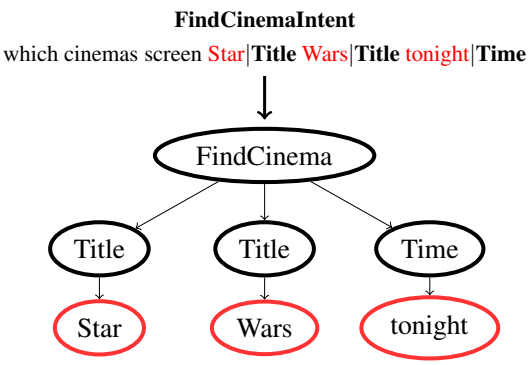

Figure 2: Conversion from intent/slot tags to tree for the sentence *Which cinemas screen Star Wars tonight?*

commercial SLU data sets. For Q&A, we consider Overnight (Wang et al., 2015b) and NLmaps (Lawrence and Riezler, 2016).

Overnight It contains sentences annotated with Lambda DCS (Liang, 2013). The sentences are divided into eight domains: *calendar*, *blocks*, *housing*, *restaurants*, *publications*, *recipes*, *socialnetwork*, and *basketball*. As shown in Table 1, the number of sentences and the terminal vocabularies are small, which makes the learning more challenging, preventing us from using data-hungry approaches such as sequence-to-sequence models.

NLmaps It contains more than two thousand questions about geographical facts, retrieved from OpenStreetMap (Haklay and Weber, 2008). Unfortunately, this data set is not divided into subdomains. While NLmaps has comparable sizes with some of the Overnight domains, its vocabularies are much larger: containing 160 terminals, 24 non-terminals and 280 word types (Table 1).

SLU We select five domains from our SLU data set: *search*, *recipes*, *cinema*, *bookings*, and *closet*. In order to investigate the use case of a new low-resource domain exploiting a higher-resource domain, we selected a mix of high-resource and low-resource domains. Details are shown in Table 1. We extracted shallow trees from data originally collected for intent/slot tagging: intents become the root of the tree, slot types are attached to the roots as their children and slot values are in turn attached to their slot types as their children. An example is shown in Figure 2. A similar approach to transform intent/slot data into tree structures has been recently employed by Gupta et al. (2018b).

DOMAIN	#	TER	NT	Words
	Q&A			
calendar	535	31	13	114
blocks	1276	30	13	99
housing	601	34	13	109
restaurants	1060	40	13	144
publications	512	24	12	80
recipes	691	30	13	121
social	2828	56	16	225
basketball	1248	40	15	148
NLmaps	1200	160	24	280
	SLU			
search	23706	1621	51	1780
recipes	18721	530	40	643
cinema	13180	806	36	923
bookings	1280	10	19	42
closet	943	63	13	107

Table 1: Details of training data. # is the number of sentences, *TER* is the terminal vocabulary size, *NT* is the non-terminal vocabulary size and *Words* is the input vocabulary size.

5 Experiments

We first run experiments on single-task semantic parsing to observe the differences among the three different data sources discussed in Section 4. Specifically, we explore the impact of an attention mechanism on the performance as well as the comparison between delexicalization and a copy mechanism for dealing with data sparsity. The metric used to evaluate parsers is the exact match accuracy, defined as the ratio of sentences correctly parsed.

5.1 Attention

Because the buffer is not consumed as in traditional transition-based parsers, Cheng et al. (2017) use an additive attention mechanism (Bahdanau et al., 2015) to focus on the more relevant words in the buffer for the current state of the stack.

In order to find the impact of attention on the different data sets, we run ablation experiments, as shown in Table 2 (left side). We found that attention between stack and buffer is not always beneficial: it appears to be helpful for larger data sets while harmful for smaller data sets. Attention is, however, useful for NLmaps, regardless of the data size. Even though NLmaps data is similarly

DOMAIN	BL	−Att	+Delex	+Copy
calendar	38.1	**43.5**	4.20	32.1
blocks	22.6	**25.1**	24.3	22.8
housing	19.0	**29.6**	6.90	21.2
restaurants	32.2	**37.3**	21.7	33.7
publications	27.3	**32.9**	11.8	26.1
recipes	47.7	**58.3**	24.1	48.1
social	44.9	**51.2**	47.7	50.9
basketball	65.2	**69.6**	38.6	66.5
NLmaps	44.9	43.5	46.4	**60.7**
search	35.6	34.9	29.2	**52.7**
recipes	40.9	37.9	37.7	**47.6**
cinema	31.5	35.5	35.7	**56.9**
bookings	72.3	77.7	72.3	**77.7**
closet	17.6	35.9	29.2	**44.1**

Table 2: Left side: Ablation experiments on attention mechanism. Right side: Comparison between delexicalization and copy mechanism. *BL* is the model of Section 2.1, *−Att* refers to the same model without attention, *+Delex* is the system with delexicalization and in *+Copy* we use a copy mechanism instead. The scores indicate the percentage of correct parses.

sized to some of the Overnight domains, its terminal space is considerably larger, perhaps making attention more important even with a smaller data set. On the other hand, the high-resource SLU's *cinema* domain is not able to benefit from the attention mechanism.

5.2 Handling Sparsity

A popular way to deal with the data sparsity problem is to delexicalize the data, that is replacing rare and unknown words with coarse categories. In our experiment, we use a named entity recognition system[2] to replace names with their named entity types. Alternatively, it is possible to use a copy mechanism to enable the decoder to copy rare words from the input rather than generating them from its limited vocabulary.

We compare the two solutions across all data sets on the right side of Table 2. Regardless of the data set, the copy mechanism generally outperforms delexicalization. We also note that delexicalization has unexpected catastrophic effects on exact match accuracy for *calendar* and *housing*. For Overnight, however, the system with copy mechanism is outperformed by the system without attention. This is unsurprising as the copy mechanism is based on attention, which is not effective on Overnight (Section 5.1). The inefficacy of copy mechanisms on the Overnight data set was also discussed in Jia and Liang (2016), where answer accuracy, rather than parsing accuracy, was used as a metric. As such, the results are not directly comparable.

For NLmaps and all SLU domains, using a copy mechanism results in an average accuracy improvement of 16% over the baseline. It is worth noting that the copy mechanism is unsurprisingly effective for SLU data due to the nature of the data set: the SLU trees were obtained from data collected for slot tagging, and as such, each leaf in the tree has to be copied from the input sentence.

Even though Overnight often yields different conclusions, most likely due to its small vocabulary size, the similar behaviors observed for NLmaps and SLU is reassuring, confirming that it is possible to unify Q&A and SLU under the same umbrella framework of executable semantic parsing.

In order to compare the NLmaps results with Lawrence and Riezler (2016), we also compute F1 scores for the data set. Our baseline outperforms previous results, achieving a score of 0.846. Our best F1 results are also obtained when adding the copy mechanism, achieving a score of 0.874.

5.3 Transfer Learning

The first set of experiments involve transfer learning across Overnight domains. For this data set, the non-terminal vocabulary is mostly shared across domains. As such, we use the architecture where only the TER output classifier is not shared. Selecting the best auxiliary domain by maximizing the overlap with the main domain was not successful, and we instead performed an exhaustive search over the domain pairs on the development set. In the interest of space, for each main domain, we report results for the best auxiliary domain (Table 3). We note that MTL and pre-training provide similar results and provide an average improvement of 4%. As expected, we observe more substantial improvements for smaller domains.

We performed the same set of experiments on the SLU domains, as shown in Table 4. In this case, the non-terminal vocabulary can vary significantly across domains. We therefore choose to use the MTL architecture where both TER and NT output classifiers are not shared. Also for SLU,

[2]https://spacy.io

DOMAIN	BL	MTL	PRETR.
calendar	43.5	**48.8**	48.2
blocks	25.1	24.1	**25.1**
housing	29.6	38.1	**38.1**
restaurants	37.3	**39.2**	36.7
publications	32.9	37.3	**40.4**
recipes	58.3	**63.4**	63.0
social	51.2	52.4	**54.5**
basketball	69.6	69.1	**71.1**

Table 3: Transfer learning results for the Overnight domains. *BL – Att* is the model without transfer learning. *PRETR.* stands for pre-training. Again, we report exact match accuracy.

DOMAIN	BL + Copy	MTL	PRETR.
search	52.7	52.3	**53.1**
cinema	56.9	**57.7**	56.4
bookings	77.7	**81.2**	78.0
closet	44.1	**52.5**	50.8

Table 4: Transfer learning results for SLU domains. *BL + Copy* is the model without transfer learning. *PRETR.* stands for pre-training. Again, the numbers are exact match accuracy.

there is no clear winner between pre-training and MTL. Nevertheless, they always outperform the baseline, demonstrating the importance of transfer learning, especially for smaller domains.

While the focus of this transfer learning framework is in exploiting high-resource domains annotated in the same way as a new low-resource domain, we also report a preliminary experiment on transfer learning across tasks. We selected the *recipes* domain, which exists in both Overnight and SLU. While the SLU data set is significantly different from Overnight, deriving from a corpus annotated with intent/slot labels, as discussed in Section 4, we found promising results using pre-training, increasing the accuracy from 58.3 to 61.1. A full investigation of transfer learning across domains belonging to heterogeneous data sets is left for future work.

The experiments on transfer learning demonstrate how parsing accuracy on low-resource domains can be improved by exploiting other domains or data sets. Except for the Overnight's *blocks* domain, which is one of the largest in Overnight, all domains in Overnight and SLU were shown to provide better results when either MTL or pre-training was used with the largest improvements observed for low-resource domains.

6 Related work

A large collection of logical forms of different nature exist in the semantic parsing literature: semantic role schemes (Palmer et al., 2005; Meyers et al., 2004; Baker et al., 1998), syntax/semantics interfaces (Steedman, 1996), executable logical forms (Liang, 2013; Kate et al., 2005), and general purpose meaning representations (Banarescu et al., 2013; Abend and Rappoport, 2013). We adopt executable logical forms in this paper. The Overnight data set uses Lambda DCS the NLmaps data set extracts meaning representations from OpenStreetMap, and the SLU data set contains logical forms reminiscent of Lambda DCS that can be used to perform actions and query databases. State-of-the-art results for the task are reported in Jia and Liang (2016); Herzig and Berant (2018). The parsers are not evaluated on the logical form they produce but on the answer they obtain using the logical form as a query. As such, their results are not directly comparable with ours.

Our semantic parsing model is an extension of the executable semantic parser of Cheng et al. (2017), which is inspired by Recurrent Neural Network Grammars (Dyer et al., 2016). We extend the model with ideas inspired by Gulcehre et al. (2016) and Luong and Manning (2016).

We build our multi-task learning architecture upon the rich literature on the topic. MTL was first introduce in Caruana (1997). It has been since used for a number of NLP problems such as tagging (Collobert and Weston, 2008), syntactic parsing (Luong et al., 2015), and machine translation (Dong et al., 2015; Luong et al., 2015). The closest to our work is Fan et al. (2017), where MTL architectures are built on top of an attentive sequence-to-sequence model (Bahdanau et al., 2015). We instead focus on transfer learning across domains of the same data sets and employ a different architecture which promises to be less data-hungry than sequence-to-sequence models.

Typical SLU systems rely on domain-specific semantic parsers that identify intents and slots in a sentence. Traditionally, these tasks were

performed by linear machine learning models (Sha and Pereira, 2003) but more recently jointly-trained DNN models are used (Mesnil et al., 2015; Hakkani-Tür et al., 2016) with differing contexts (Gupta et al., 2018a; Vishal Ishwar Naik, 2018). More recently there has been work on extending the traditional intent/slot framework using targeted parsing to handle more complex linguistic phenomenon like coordination (Gupta et al., 2018c; Agarwal et al., 2018).

7 Conclusions

We framed SLU as an executable semantic parsing task, which addresses a limitation of current commercial SLU systems. By applying our framework to different data sets, we demonstrate that the framework is effective for Q&A as well as for SLU. We explored a typical scenario where it is necessary to learn a semantic parser for a new domain with little data, but other high-resource domains are available. We show the effectiveness of our system by achieving state-of-the-art parsing performance on NLmaps and strong baselines on Overnight, and the effectiveness of both pre-training and MTL on different domains and data sets. Preliminary experiment results on transfer learning across domains belonging to heterogeneous data sets suggest future work in this area.

Acknowledgments

The authors would like to thank the three anonymous reviewers for their comments and the Amazon Alexa NLU team members for their feedback.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of ACL*.

Sanchit Agarwal, Rahul Goel, Tagyoung Chung, Abhishek Sethi, Arindam Mandal, and Spyros Matsoukas. 2018. Parsing coordination for spoken language understanding. *arXiv preprint arXiv:1810.11497*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of COLING*.

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. *Linguistic Annotation Workshop*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 44–55.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of CoNLL*. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of ACL*.

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 ACL*.

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

Alex Graves. 2012. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *Proceedings of ACL*.

Raghav Gupta, Abhinav Rastogi, and Dilek Hakkani-Tur. 2018a. An efficient approach to encoding context for spoken language understanding. *arXiv preprint arXiv:1807.00267*.

Sonal Gupta, Rushin Shah, Mrinal Mohit, and Anuj Kumar. 2018b. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of EMNLP*.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018c. Semantic parsing for task oriented dialog using hierarchical representations. *arXiv preprint arXiv:1810.07942*.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.

Mordechai Haklay and Patrick Weber. 2008. Openstreetmap: User-generated street maps. *Ieee Pervas Comput*, 7(4):12–18.

Jonathan Herzig and Jonathan Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. *Proceedings of EMNLP*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 12–22.

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1062. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Carolin Lawrence and Stefan Riezler. 2016. Nlmaps: A natural language interface to query openstreetmap. In *Proceedings of COLING*.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*.

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

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the ACL*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. Annotating noun argument structure for nombank. In *Proceedings of LREC*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*. Citeseer.

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

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics.

Mark Steedman. 1996. Surface structure and interpretation.

Sebastian Thrun. 1996. Is learning the n-th thing any easier than learning the first? In *Proceedings of NIPS*.

Rahul Goel Vishal Ishwar Naik, Angeliki Metallinou. 2018. Context aware conversational understanding for intelligent agents with a screen.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. A transition-based algorithm for amr parsing. In *Proceedings of NAACL*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015b. Building a semantic parser overnight. In *Proceedings of ACL*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of EMNLP*.

This figure "arch.png" is available in "png" format from:

http://arxiv.org/ps/1903.04521v2