Transfer Learning for Neural Semantic Parsing

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer
Amazon.com
{fanxing, mathias, mddreyer}@amazon.com, monti@amazon.co.uk

Abstract

The goal of semantic parsing is to map natural language to a machine interpretable meaning representation language (MRL). One of the constraints that limits full exploration of deep learning technologies for semantic parsing is the lack of sufficient annotation training data. In this paper, we propose using sequence-to-sequence in a multi-task setup for semantic parsing with a focus on transfer learning. We explore three multi-task architectures for sequence-to-sequence modeling and compare their performance with an independently trained model. Our experiments show that the multi-task setup aids transfer learning from an auxiliary task with large labeled data to a target task with smaller labeled data. We see absolute accuracy gains ranging from 1.0% to 4.4% in our in-house data set, and we also see good gains ranging from 2.5% to 7.0% on the ATIS semantic parsing tasks with syntactic and semantic auxiliary tasks.

1 Introduction

Conversational agents, such as Alexa, Siri and Cortana, solve complex tasks by interacting and mediating between the end-user and multiple backend software applications and services. Natural language is a simple interface used for communication between these agents. However, to make natural language machine-readable we need to map it to a representation that describes the semantics of the task expressed in the language. Semantic parsing is the process of mapping a natural-language sentence into a formal machine-readable representation of its meaning. This poses a challenge in a multi-tenant system that has to interact with multiple backend knowledge sources each with their own semantic formalisms and custom schemas for accessing information, where each formalism has various amount of annotation training data.

Recent works have proven sequence-to-sequence to be an effective model architecture (Jia and Liang, 2016; Dong and Lapata, 2016) for semantic parsing. However, because of the limit amount of annotated data, the advantage of neural networks to capture complex data representation using deep structure (Johnson et al., 2016) has not been fully explored. Acquiring data is expensive and sometimes infeasible for task-oriented systems, the main reasons being multiple formalisms (e.g., SPARQL for WikiData (Vrandečić and Krötzsch, 2014), MQL for Freebase (Flanagan, 2008)), and multiple tasks (question answering, navigation interactions, transactional interactions). We propose to exploit these multiple representations in a multi-task framework so we can minimize the need for a large labeled corpora across these formalisms. By suitably modifying the learning process, we capture the common structures that are implicit across these formalisms and the tasks they are targeted for.

In this work, we focus on a sequence-to-sequence based transfer learning for semantic parsing. In order to tackle the challenge of multiple formalisms, we apply three multi-task frameworks with different levels of parameter sharing. Our hypothesis is that the encoder-decoder paradigm learns a canonicalized representation across all tasks. Over a strong single-task sequence-to-sequence baseline, our proposed approach shows accuracy improvements across the target formalism. In addition, we show that even when the auxiliary task is syntactic parsing we can achieve good gains in semantic parsing that are comparable to the published state-of-the-art.

Proceedings of the 2nd Workshop on Representation Learning for NLP, pages 48–56,
Vancouver, Canada, August 3, 2017.

2 Related Work

There is a large body of work for semantic parsing. These approaches fall into three broad categories – completely supervised learning based on fully annotated logical forms associated with each sentence (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2012) using question-answer pairs and conversation logs as supervision (Artzi and Zettlemoyer, 2011; Liang et al., 2011; Berant et al., 2013) and distant supervision (Cai and Yates, 2013; Reddy et al., 2014). All these approaches make assumptions about the task, features and the target semantic formalism.

On the other hand, neural network based approaches, in particular the use of recurrent neural networks (RNNs) and encoder-decoder paradigms (Sutskever et al., 2014), have made fast progress on achieving state-of-the art performance on various NLP tasks (Vinyals et al., 2015; Dyer et al., 2015; Bahdanau et al., 2014). A key advantage of RNNs in the encoder-decoder paradigm is that very few assumptions are made about the domain, language and the semantic formalism. This implies they can generalize faster with little feature engineering.

Full semantic graphs can be expensive to annotate, and efforts to date have been fragmented across different formalisms, leading to a limited amount of annotated data in any single formalism. Using neural networks to train semantic parsers on limited data is quite challenging. Multi-task learning aims at improving the generalization performance of a task using related tasks (Caruana, 1998; Ando and Zhang, 2005; Smith and Smith, 2004). This opens the opportunity to utilize large amounts of data for a related task to improve the performance across all tasks. There has been recent work in NLP demonstrating improved performance for machine translation (Dong et al., 2015) and syntactic parsing (Luong et al., 2015).

In this work, we attempt to merge various strands of research using sequence-to-sequence modeling for semantic parsing with focusing on improving semantic formalisms with small amount of training data using a multi-task model architecture. The closest work is Herzig and Berant (2017). Similar to this work, the authors use a neural semantic parsing model in a multi-task framework to jointly learn over multiple knowledge bases. Our work differs from their work in that we focus our attention on transfer learning, where we have access to a large labeled resource in one task and want another semantic formalism with access to limited training data to benefit from a multi-task learning setup. Furthermore, we also demonstrate that we can improve semantic parsing tasks by using large data sources from an auxiliary task such as syntactic parsing, thereby opening up the opportunity for leveraging much larger datasets. Finally, we carefully compare multiple multi-task architectures in our setup and show that increased sharing of both the encoder and decoder along with shared attention results in the best performance.

3 Problem Formulation

3.1 Sequence-to-Sequence Formulation

Our semantic parser extends the basic encoder-decoder approach in Jia and Liang (2016). Given a sequence of inputs $\boldsymbol{x} = x_1, \ldots, x_m$, the sequence-to-sequence model will generate an output sequence of $\boldsymbol{y} = y_1, \ldots, y_n$. We encode the input tokens $\boldsymbol{x} = x_1, \ldots, x_m$ into a sequence of embeddings $\boldsymbol{h} = \boldsymbol{h}_1, \ldots, \boldsymbol{h}_m$

$$\boldsymbol{h}_i = f_{\text{encoder}}(E_x(x_i), \boldsymbol{h}_{i-1}) \quad (1)$$

First, an input embedding layer E_x maps each word x_i to a fixed-dimensional vector which is then fed as input to the network f to obtain the hidden state representation $\boldsymbol{h}_i$. The embedding layer E_x could contain one single word embedding lookup table or a combination of word and gazetteer embeddings, where we concatenate the output from each table. For the encoder and decoder, we use a stacked Gated Recurrent Units (GRU) (Cho et al., 2014).[1] The hidden states are then converted to one fixed-length context vector per output index, $\boldsymbol{c}_j = \phi_j(h_1, \ldots, h_m)$, where ϕ_j summarizes all input hidden states to form the context for a given output index j.[2]

The decoder then uses these fixed-length vectors $\boldsymbol{c}_j$ to create the target sequence through the following model. At each time step j in the output sequence, a state $\boldsymbol{s}_j$ is calculated as

$$\boldsymbol{s}_j = f_{\text{decoder}}(E_y(y_{j-1}), \boldsymbol{s}_{j-1}, \boldsymbol{c}_j) \quad (2)$$

[1] In order to speedup training, we use a right-to-left GRU instead of a bidirectional GRU.

[2] In a vanilla decoder, each $\phi_j(h_1, \ldots, h_m) \stackrel{\text{def}}{=} h_m$, i.e, the hidden representation from the last state of the encoder is used as context for every output time step j.

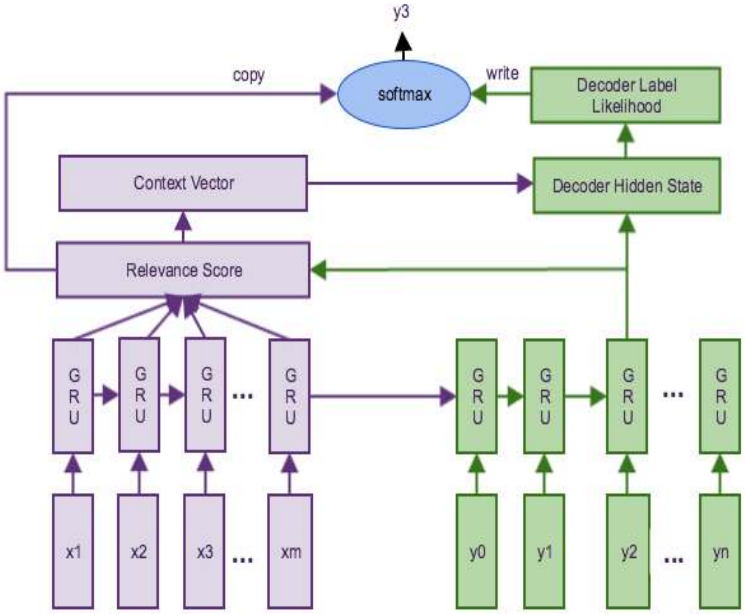


Figure 1: An example of how the decoder output y_3 is generated.

Here, E_y maps any output symbol to a fixed-dimensional vector. Finally, we compute the probability of the output symbol y_j given the history $y_{<j}$ using Equation 3.

$$p(y_j \mid y_{<j}, \boldsymbol{x}) \propto \exp(\boldsymbol{O}[\boldsymbol{s}_j; \boldsymbol{c}_j]) \tag{3}$$

where the matrix $\boldsymbol{O}$ projects the concatenation of $\boldsymbol{s}_j$ and $\boldsymbol{c}_j$, denoted as $[\boldsymbol{s}_j; \boldsymbol{c}_j]$, to the final output space. The matrix $\boldsymbol{O}$ are part of the trainable model parameters. We use an attention mechanism (Bahdanau et al., 2014) to summarize the context vector $\boldsymbol{c}_j$,

$$\boldsymbol{c}_j = \phi_j(h_1, \ldots, h_m) = \sum_{i=1}^{m} \alpha_{ji}\, \boldsymbol{h}_i \tag{4}$$

where $j \in [1, \ldots, n]$ is the step index for the decoder output and α_{ji} is the attention weight, calculated using a softmax:

$$\alpha_{ji} = \frac{\exp(e_{ji})}{\sum_{i'=1}^{m} \exp(e_{ji'})} \tag{5}$$

where e_{ji} is the relevance score of each context vector $\boldsymbol{c}_j$, modeled as:

$$e_{ji} = g(\boldsymbol{h}_i, \boldsymbol{s}_j) \tag{6}$$

In this paper, the function g is defined as follows:

$$g(\boldsymbol{h}_i, \boldsymbol{s}_j) = \boldsymbol{v}^\intercal \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}_j) \tag{7}$$

where $\boldsymbol{v}$, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are trainable parameters.

In order to deal with the large vocabularies in the output layer introduced by the long tail of entities in typical semantic parsing tasks, we use a copy mechanism (Jia and Liang, 2016). At each time step j, the decoder chooses to either copy a token from the encoder's input stream or to write a token from the the decoder's fixed output vocabulary. We define two actions:

1. WRITE$[y]$ for some $y \in \mathcal{V}_{\text{decoder}}$, where $\mathcal{V}_{\text{decoder}}$ is the output vocabulary of the decoder.

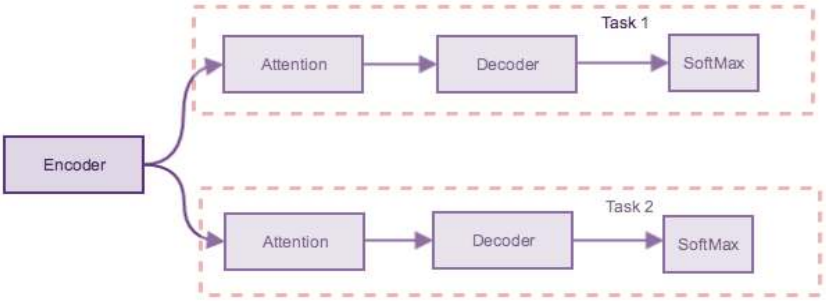
2. COPY$[i]$ for some $i \in 1, \ldots, m$, which copies one symbol from the m input tokens.

We formulate a single softmax to select the action to take, rewriting Equation 3 as follows:

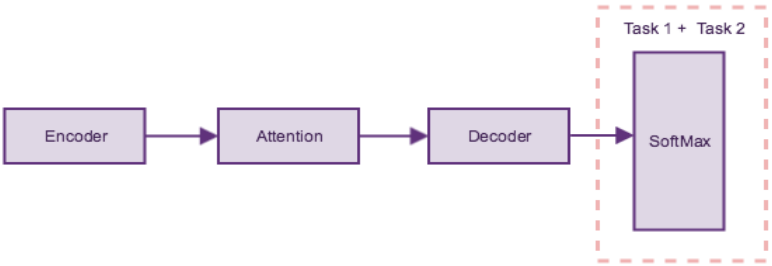
$$p(a_j = \text{WRITE}[y_j] \mid y_{<j}, \boldsymbol{x}) \propto \exp(\boldsymbol{O}[\boldsymbol{s}_j; \boldsymbol{c}_j]) \tag{8}$$

$$p(a_j = \text{COPY}[i] \mid y_{<j}, \boldsymbol{x}) \propto \exp(e_{ji}) \tag{9}$$

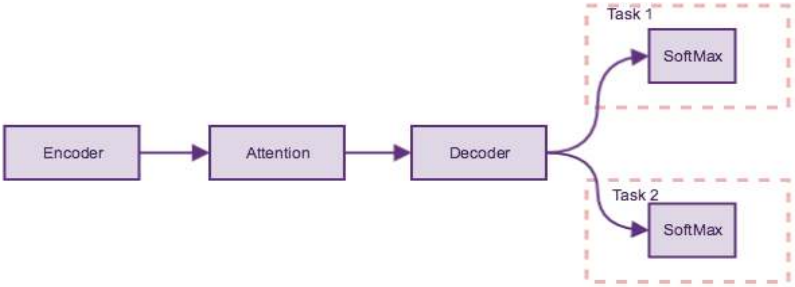
The decoder is now a softmax over the actions a_j; Figure 1 shows how the decoder's output $\boldsymbol{y}$ at the third time step y_3 is generated. At each time step, the decoder will make a decision to copy a particular token from input stream or to write a token from the fixed output label pool.



(a) *one-to-many*: A multi-task architecture where only the encoder is shared across the two tasks.



(b) *one-to-one*: A multi-task architecture where both the encoder and decoder along with the attention layer are shared across the two tasks.



(c) *one-to-shareMany*: A multi-task architecture where both the encoder and decoder along with the attention layer are shared across the two tasks, but the final softmax output layer is trained differently, one for each task.

Figure 2: Three multi-task architectures.

3.2 Multi-task Setup

We focus on training scenarios where multiple training sources K are available. Each source K can be considered a domain or a task, which consists of pairs of utterance $\boldsymbol{x}$ and annotated logical form $\boldsymbol{y}$. There are no constraints on the logical forms having the same formalism across the K domains. Also, the tasks K can be different, e.g., we can mix semantic parsing and syntactic parsing tasks. We also assume that given an utterance, we already know its associated source K in both training and testing.

In this work, we explore and compare three multi-task sequence-to-sequence model architectures: one-to-many, one-to-one and one-to-shareMany.

3.2.1 One-to-Many Architecture

This is the simplest extension of sequence-to-sequence models to the multi-task case. The encoder is shared across all the K tasks, but the decoder and attention parameters are not shared. The shared encoder captures the English language sequence, whereas each decoder is trained to predict its own formalism. This architecture is shown in Figure 2a. For each minibatch, we uniformly sample among all training sources, choosing one source to select data exclusively from. Therefore, at each model parameter update, we only update the encoder, attention module and the decoder for the selected source, while the parameters for the

other $K-1$ decoder and attention modules remain the same.

3.2.2 One-to-One Architecture

Figure 2b shows the one-to-one architecture. Here we have a single sequence-to-sequence model across all the tasks, i.e., the embedding, encoder, attention, decoder and the final output layers are shared across all the K tasks. In this architecture, the number of parameters is independent of the number of tasks K. Since there is no explicit representation of the domain/task that is being decoded, the input is augmented with an artificial token at the start to identify the task the same way as in Johnson et al. (2016).

3.2.3 One-to-ShareMany Architecture

We show the model architecture for one-to-shareMany in Figure 2c. The model modifies the one-to-many model by encouraging further sharing of the decoder weights. Compared with the one-to-one model, the one-to-shareMany differs in the following aspects:

- Each task has its own output layer. Our hypothesis is that by separating the tasks in the final layer we can still get the benefit of sharing the parameters, while fine-tuning for specific tasks in the output, resulting in better accuracy on each individual task.

- The one-to-one requires a concatenation of all output labels from training sources. During training, every minibatch needs to be forwarded and projected to this large softmax layer. While for one-to-ShareMany, each minibatch just needs to be fed to the softmax associated with the chosen source. Therefore, the one-to-shareMany is faster to train especially in cases where the output label size is large.

- The one-to-one architecture is susceptible to data imbalance across the multiple tasks, and typically requires data upsampling or downsampling. While for one-to-shareMany we can alternate the minibatches amongst the K sources using uniform selection.

  From the perspective of neural network optimization, mixing the small training data with a large data set from the auxiliary task can be also seen as adding noise to the training process and hence be helpful for generalization and to avoid overfitting. With the auxiliary tasks, we are able to train large size modesl that can handle complex task without worrying about overfitting.

4 Experiments

4.1 Data Setup

We mainly consider two Alexa dependency-based semantic formalisms in use – an Alexa meaning representation language (AlexaMRL), which is a lightweight formalism used for providing built-in functionality for developers to develop their own skills.[3] The other formalism we consider is the one used by Evi,[4] a question-answering system used in Alexa. Evi uses a proprietary formalism for semantic understanding; we will call this the Evi meaning representation language (EviMRL). Both these formalisms aim to represent natural language. While the EviMRL is aligned with an internal schema specific to the knowledge base (KB), the AlexaMRL is aligned with an RDF-based open-source ontology (Guha et al., 2016). Figure 3 shows two example utterances and their parses in both EviMRL and AlexaMRL formalisms.

Our training set consists of $200K$ utterances – a fraction of our production data, annotated using AlexaMRL – as our main task. For the EviMRL task, we have $> 1M$ utterances data set for training. We use a test set of $30K$ utterances for AlexaMRL testing, and $366K$ utterances for EviMRL testing. To show the effectiveness of our proposed method, we also use the ATIS corpora as the small task for our transfer learning framework, which has 4480 training and 448 test utterances (Zettlemoyer and Collins, 2007). We also include an auxiliary task such as syntactic parsing in order to demonstrate the flexibility of the multi-task paradigm. We use $34K$ WSJ training data for syntactic constituency parsing as the large task, similar to the corpus in Vinyals et al. (2015).

We use Tensorflow (Abadi et al., 2016) in all our experiments, with extensions for the copy mechanism. Unless stated otherwise, we train all models for 10 epochs, with a fixed learning rate of 0.5 for the first 6 epochs and halve it subsequently for every epoch. The mini-batch size used is 128. The encoder and decoder use a 3-layer GRU with 512

[3] For details see https://tinyurl.com/lnfh9py.

[4] https://www.evi.com

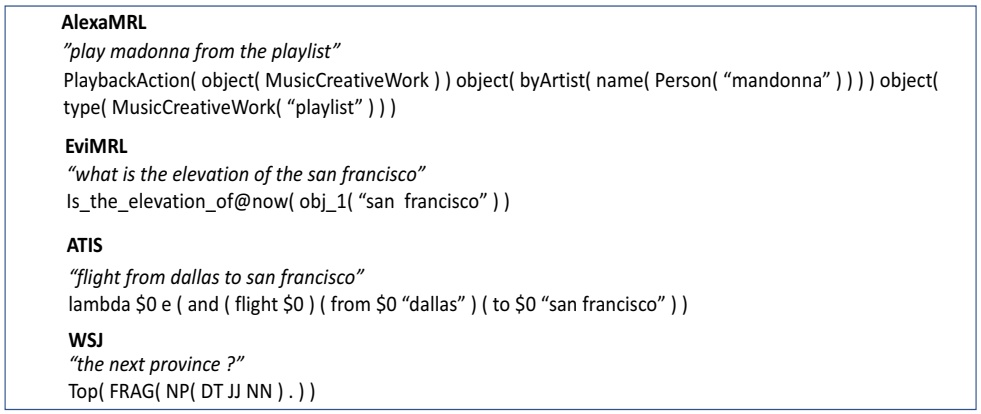


Figure 3: Example utterances for the multiple semantic formalisms

hidden units. We apply dropout with probability of 0.2 during training. All models are initialized with pre-trained 300-dimension GloVe embeddings (Pennington et al., 2014). We also apply label embeddings with 300 dimension for the output labels that are randomly initialized and learned during training. The input sequence is reversed before sending it to the encoder (Vinyals et al., 2015). We use greedy search during decoding. The output label size for EviMRL is $2K$ and for Alexa is < 100. For the multi-task setup, we use a vocabulary size of about $50K$, and for AlexaMRL independent task, we use a vocabulary size of about $20K$. We post-process the output of the decoder by balancing the brackets and determinizing the units of production to avoid duplicates.

4.2 AlexaMRL Transfer Learning Experiments

We first study the effectiveness of the multi-task architecture in a transfer learning setup. Here we consider EviMRL as the large source auxiliary task and the AlexaMRL as the target task we want to transfer learn. We consider various data sizes for the target task – $10K$, $50K$ and $100K$ and $200K$ by downsampling. For each target data size, we compare a single-task setup, trained on the target task only, with the the various multi-task setups from Section 3.2 – independent, one-to-one, one-to-many, and one-to-manyShare. Figure 4 summarizes the results. The x-axis lists the four model architecture, and y-axis is the accuracy. The positive number above the mark of one-to-one, one-to-many and one-to-manyShare represents the absolute accuracy gain compared with the independent model. For the $10k$ independent model, we reduce the hidden layer size from 512 to 256 to optimize the performance.

In all cases, the multi-task architectures provide accuracy improvements over the independent architecture. By jointly training across the two tasks, the model is able to leverage the richer syntactic/semantic structure of the larger task (EviMRL), resulting in an improved encoding of the input utterance that is then fed to the decoder resulting in improved accuracy over the smaller task (AlexaMRL).

We take this sharing further in the one-to-one and one-to-shareMany architecture by introducing shared decoder parameters, which forces the model to learn a common canonical representation for solving the semantic parsing task. Doing so, we see further gains across all data sizes in 4. For instance, in the 200k case, the absolute gain improves from $+2.0$ to $+2.7$. As the training data size for the target task increases, we tend to see relatively less gain from model sharing. For instance, in 10k training cases, the absolute gain from the one-to-one and one-to-manyshared is 1.6, this gain reduces to 0.7 when we have 200k training data.

When we have a small amount of training data, the one-to-shareMany provides better accuracy compared with one-to-one. For instance, we see 1.0 and 0.6 absolute gain from one-to-one to one-to-shareMany for 10k and 50k cases respectively. However, no gain is observed for 100k and 200k training cases. This confirms the hypothesis that for small amounts of data, having a dedicated output layer is helpful to guide the training.

Transfer learning works best when the source data is large, thereby allowing the smaller task to leverage the rich representation of the larger task.

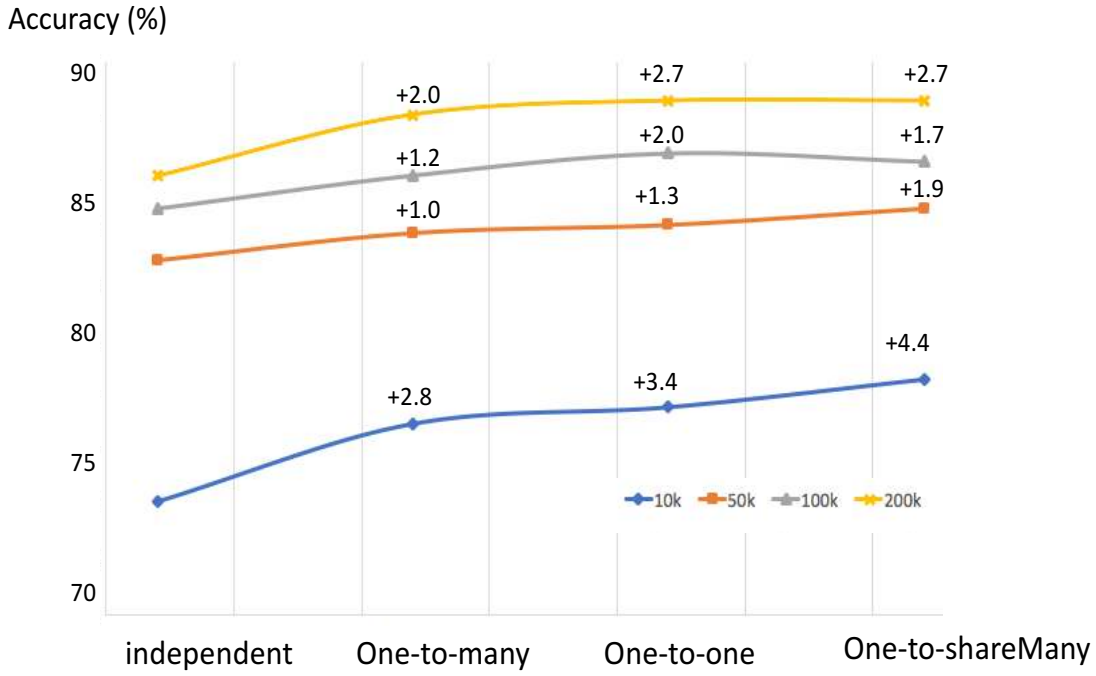


Figure 4: *Accuracy for AlexaMRL.*

However, as the training data size increases, the accuracy gains from the shared architectures become smaller – the largest gain of 4.4% absolute is observed in the $10K$ setting, but as the data increases to $200K$ the improvements are almost halved to about 2.7%.

In Table 1, we summarize the numbers of parameters in each of the four model architectures and their step time.[5] As expected, we see comparable training time for one-to-many and one-to-shareMany, but 10% step time increase for one-to-one. We also see that one-to-one and one-to-shareMany have similar number of parameter, which is about 15% smaller than one-to-many due to the sharing of weights. The one-to-shareMany architecture is able to get the increased sharing while still maintaining reasonable training speed per step-size.

We also test the accuracy of EviMRL with the transfer learning framework. To our surprise, the EviMRL task also benefits from the AlexMRL task. We observe an absolute increase of accuracy of 1.3% over the EviMRL baseline.[6] This observation reinforces the hypothesis that combining data from different semantic formalisms helps the generalization of the model by capturing common sub-structures involved in solving semantic parsing tasks across multiple formalisms.

Model architecture	**param. size**	**step time**
independent	15 million	0.51
one-to-many	33 million	0.66
one-to-one	28 million	0.71
one-to-shareMany	28 million	0.65

Table 1: parameter size and training time comparision for independent and multi-task models

4.3 Transfer Learning Experiments on ATIS

Here, we apply the described transfer learning setups to the ATIS semantic parsing task (Zettlemoyer and Collins, 2007). We use a single GRU layer of 128 hidden states to train the independent model. During transfer learning, we increase the model size to two hidden layers each with 512 hid-

[5] In our experiment, it is the training time for a 128 size minibatches update on Nvidia Tesla K80 GPU

[6] The baseline is at 90.9% accuracy for the single task sequence-to-sequence model

den states. We adjust the minibatch size to 20 and dropout rate to 0.2 for independent model and 0.7 for multi-task model. We post-process the model output, balancing the braces and removing duplicates in the output. The initial learning rate has been adjusted to 0.8 using the dev set. Here, we only report accuracy numbers for the independent and one-to-shareMany frameworks. Correctness is based on denotation match at utterance level. We summarize all the results in Table 2.

System	Test accuracy
Previous work	
Zettlemoyer and Collins (2007)	**84.6**
Kwiatkowski et al. (2011)	82.8
Poon (2013)	83.5
Zhao and Huang (2014)	84.2
Jia and Liang (2016)	83.3
Dong and Lapata (2016)	84.2
Our work	
Independent model	77.2
+ WSJ constituency parsing	79.7
+ EviMRL semantic parsing	84.2

Table 2: Accuracy on ATIS

Our independent model has an accuracy of 77.2%, which is comparable to the published baseline of 76.3% reported in Jia and Liang (2016) before their data recombination. To start with, we first consider using a related but complementary task – syntactic constituency parsing, to help improve the semantic parsing task. By adding WSJ constituency parsing as an auxiliary task for ATIS, we see a 3% relative improvement in accuracy over the independent task baseline. This demonstrates that the multi-task architecture is quite general and is not constrained to using semantic parsing as the auxiliary task. This is important as it opens up the possibility of using significantly larger training data on tasks where acquiring labels is relatively easy.

We then add the EviMRL data of $> 1M$ instances to the multi-task setup as a third task, and we see further relative improvement of 5%, which is comparable to the published state of the art (Zettlemoyer and Collins, 2007) and matches the neural network setup in Dong and Lapata (2016).

5 Conclusion

We presented sequence-to-sequence architectures for transfer learning applied to semantic parsing. We explored multiple architectures for multi-task decoding and found that increased parameter sharing results in improved performance especially when the target task data has limited amounts of training data. We observed a 1.0-4.4% absolute accuracy improvement on our internal test set with 10k-200k training data. On ATIS, we observed a $> 6\%$ accuracy gain.

The results demonstrate the capabilities of sequence-to-sequence modeling to capture a canonicalized representation between tasks, particularly when the architecture uses shared parameters across all its components. Furthermore, by utilizing an auxiliary task like syntactic parsing, we can improve the performance on the target semantic parsing task, showing that the sequence-to-sequence architecture effectively leverages the common structures of syntax and semantics. In future work, we want to use this architecture to build models in an incremental manner where the number of sub-tasks K continually grows. We also want to explore auxiliary tasks across multiple languages so we can train multilingual semantic parsers simultaneously, and use transfer learning to combat labeled data sparsity.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research* 6(Nov):1817–1853.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 421–432.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase

from question-answer pairs. In *EMNLP*. volume 2, page 6.

Qingqing Cai and Alexander Yates. 2013. Semantic parsing Freebase: Towards open-domain semantic parsing. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*. volume 1, pages 328–338.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer, pages 95–133.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL (1)*. pages 1723–1732.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280* .

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075* .

David Flanagan. 2008. Mql reference guide. *Metaweb Technologies, Inc* page 2.

Ramanathan V Guha, Dan Brickley, and Steve Macbeth. 2016. Schema. org: Evolution of structured data on the web. *Communications of the ACM* 59(2):44–51.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. *https://arxiv.org/abs/1702.01569* .

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622* .

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558* .

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1512–1523.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 590–599.

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* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *ACL (1)*. Citeseer, pages 933–943.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics* 2:377–392.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using english to parse korean. In *EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM* 57(10):78–85.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*. pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *EMNLP*. pages 678–687.

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* .

Kai Zhao and Liang Huang. 2014. Type-driven incremental semantic parsing with polymorphism. *arXiv preprint arXiv:1411.5379* .