Neural Semantic Parsing over Multiple Knowledge-bases

Jonathan Herzig[1,2] and **Jonathan Berant**[1]

[1]Tel-Aviv University, Tel Aviv-Yafo, Israel
[2]IBM Research, Haifa 31905, Israel
jherzig@gmail.com, joberant@cs.tau.ac.i

Abstract

A fundamental challenge in developing semantic parsers is the paucity of strong supervision in the form of language utterances annotated with logical form. In this paper, we propose to exploit structural regularities in language in different domains, and train semantic parsers over multiple knowledge-bases (KBs), while sharing information across datasets. We find that we can substantially improve parsing accuracy by training a single sequence-to-sequence model over multiple KBs, when providing an encoding of the domain at decoding time. Our model achieves state-of-the-art performance on the OVERNIGHT dataset (containing eight domains), improves performance over a single KB baseline from 75.6% to 79.6%, while obtaining a 7x reduction in the number of model parameters.

1 Introduction

Semantic parsing is concerned with translating language utterances into executable logical forms and constitutes a key technology for developing conversational interfaces (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2011; Liang et al., 2011; Artzi and Zettlemoyer, 2013; Berant and Liang, 2015).

A fundamental obstacle to widespread use of semantic parsers is the high cost of annotating logical forms in new domains. To tackle this problem, prior work suggested strategies such as training from denotations (Clarke et al., 2010; Liang et al., 2011; Artzi and Zettlemoyer, 2013), from paraphrases (Berant and Liang, 2014; Wang et al., 2015) and from declarative sentences (Krishnamurthy and Mitchell, 2012; Reddy et al., 2014).

Example 1
Domain: HOUSING
*"Find a housing that is **no more than** 800 square feet."*,
$\texttt{Type.HousingUnit} \sqcap \texttt{Size.} \leq .800$
Domain: PUBLICATIONS
*"Find an article with **no more than** two authors"*
$\texttt{Type.Article} \sqcap \mathbf{R}[\lambda x.\text{count}(\texttt{AuthorOf}.x)].\leq .2$

Example 2
Domain: RESTAURANTS
*"which restaurant has **the most** ratings?"*
$\text{argmax}(\texttt{Type.Restaurant}, \mathbf{R}[\lambda x.\text{count}(\mathbf{R}[\texttt{Rating}].x)])$
Domain: CALENDAR
*"which meeting is attended by **the most** people?"*
$\text{argmax}(\texttt{Type.Meeting}, \mathbf{R}[\lambda x.\text{count}(\mathbf{R}[\texttt{Attendee}].x)])$

Figure 1: Examples for natural language utterances with logical forms in lambda-DCS (Liang, 2013) in different domains that share structural regularity (a comparative structure in the first example and a superlative in the second).

In this paper, we suggest an orthogonal solution: to pool examples from multiple datasets in different domains, each corresponding to a separate knowledge-base (KB), and train a model over all examples. This is motivated by an observation that while KBs differ in their entities and properties, the structure of language composition repeats across domains (Figure 1). E.g., a superlative in language will correspond to an 'argmax', and a verb followed by a noun often denotes a join operation. A model that shares information across domains can improve generalization compared to a model that is trained on a single domain only.

Recently, Jia and Liang (2016) and Dong and Lapata (2016) proposed sequence-to-sequence models for semantic parsing. Such neural models substantially facilitate information sharing, as both language and logical form are represented with similar abstract vector representations in all domains. We build on their work and examine models that share representations across domains during encoding of language and decoding of logical form, inspired by work on domain adaptation

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 623–628
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-2098

(Daume III, 2007) and multi-task learning (Caruana, 1997; Collobert et al., 2011; Luong et al., 2016; Firat et al., 2016; Johnson et al., 2016). We find that by providing the decoder with a representation of the domain, we can train a single model over multiple domains and substantially improve accuracy compared to models trained on each domain separately. On the OVERNIGHT dataset, this improves accuracy from 75.6% to 79.6%, setting a new state-of-the-art, while reducing the number of parameters by a factor of 7. To our knowledge, this work is the first to train a semantic parser over multiple KBs.

2 Problem Setup

We briefly review the model presented by Jia and Liang (2016), which we base our model on.

Semantic parsing can be viewed as a sequence-to-sequence problem (Sutskever et al., 2014), where a sequence of input language tokens $x = x_1, \ldots, x_m$ is mapped to a sequence of output logical tokens $y_1, \ldots, y_n$.

The **encoder** converts $x_1, \ldots, x_m$ into a sequence of context sensitive embeddings $b_1, \ldots, b_m$ using a bidirectional RNN (Bahdanau et al., 2015): a forward RNN generates hidden states $h_1^F, \ldots, h_m^F$ by applying the LSTM recurrence (Hochreiter and Schmidhuber, 1997): $h_i^F = LSTM(\phi^{(in)}(x_i), h_{i-1}^F)$, where $\phi^{(in)}$ is an embedding function mapping a word x_i to a fixed-dimensional vector. A backward RNN similarly generates hidden states $h_m^B, \ldots, h_1^B$ by processing the input sequence in reverse. Finally, for each input position i, the representation b_i is the concatenation $[h_i^F, h_i^B]$. An attention-based **decoder** (Bahdanau et al., 2015; Luong et al., 2015) generates output tokens one at a time. At each time step j, it generates y_j based on the current hidden state s_j, then updates the hidden state s_{j+1} based on s_j and y_j. Formally, the decoder is defined by the following equations:

$$
\begin{aligned}
s_1 &= \tanh(W^{(s)}[h_m^F, h_1^B]), \\
e_{ji} &= s_j^\top W^{(a)} b_i, \\
\alpha_{ji} &= \frac{\exp(e_{ji})}{\sum_{i'=1}^{m} e_{ji'}}, \\
c_j &= \sum_{i=1}^{m} \alpha_{ji} b_i, \\
p(y_j = w \mid x, y_{1:j-1}) &\propto \exp(U[s_j, c_j]), \\
s_{j+1} &= LSTM([\phi^{(out)}(y_j), c_j], s_j),
\end{aligned}
\tag{1}
$$

where $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$. The matrices $W^{(s)}$, $W^{(a)}$, U, and the embedding function $\phi^{(out)}$ are decoder parameters. We also employ attention-based copying as described by Jia and Liang (2016), but omit details for brevity.

The entire model is trained end-to-end by maximizing $p(y \mid x) = \prod_{j=1}^{n} p(y_j \mid x, y_{1:j-1})$.

3 Models over Multiple KBs

In this paper, we focus on a setting where we have access to K training sets from different domains, and each domain corresponds to a different KB. In all domains the input is a language utterance and the label is a logical form (we assume annotated logical forms can be converted to a single formal language such as lambda-DCS in Figure 1). While the mapping from words to KB constants is specific to each domain, we expect that the manner in which language expresses composition of meaning to be shared across domains. We now describe architectures that share information between the encoders and decoders of different domains.

3.1 One-to-one model

This model is similar to the baseline model described in Section 2. As illustrated in Figure 2, it consists of a single encoder and a single decoder, which are used to generate outputs for all domains. Thus, all model parameters are shared across domains, and the model is trained from all examples. Note that the number of parameters does not depend on the number of domains K.

Since there is no explicit representation of the domain that is being decoded, the model must learn to identify the domain given only the input. To alleviate that, we encode the k'th domain by a one-hot vector $d_k \in \mathbb{R}^K$. At each step, the decoder updates the hidden state conditioned on the domain's one-hot vector, as well as on the previous hidden state, the output token and the context. Formally, for domain k, Equation 1 is changed:[1]

$$
s_{j+1} = LSTM([\phi^{(out)}(y_j), c_j, d_k], s_j). \tag{2}
$$

Recently Johnson et al. (2016) used a similar intuition for neural machine translation, where they added an artificial token at the beginning of each source sentence to specify the target language. We implemented their approach and compare to it in Section 4.

[1] For simplicity, we omit the domain index k from our notation whenever it can be inferred from context.

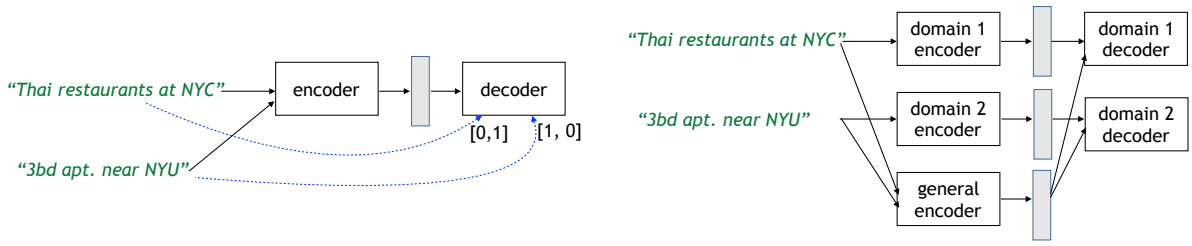

Figure 2: Illustration of models with one example from the RESTAURANTS domain and another from the HOUSING domain. Left: One-to-one model with optional domain encoding. Right: many-to-many model.

Since we have one decoder for multiple domains, tokens which are not in the domain vocabulary could possibly be generated. We prevent that at test time by excluding out-of-domain tokens before the softmax ($p(y_j \mid x, y_{1:j-1})$) takes place.

3.2 Many-to-many model

In this model, we keep a separate encoder and decoder for every domain, but augment the model with an additional encoder that consumes examples from all domains (see Figure 2). This is motivated by prior work on domain adaptation (Daume III, 2007; Blitzer et al., 2011), where each example has a representation that captures domain-specific aspects of the example and a representation that captures domain-general aspects. In our case, this is achieved by encoding examples with a domain-specific encoder as well as a domain-general encoder, and passing both representations to the decoder.

Formally, we now have $K + 1$ encoders and K decoders, and denote by $h_i^{F,k}, h_i^{B,k}, b_i^k$ the forward state, backward state and their concatenation at position i (the domain-general encoder has index $K + 1$). The hidden state of the decoder in domain k is initialized from the domain-specific and domain-general encoder:

$$s_1 = \tanh(W^{(s)}[h_m^{F,k}, h_1^{B,k}, h_m^{F,K+1}, h_1^{B,K+1}]).$$

Then, we compute unnormalized attention scores based on both encoders, and represent the language context with both domain-general and domain-specific representations. Equation 1 for domain k is changed as follows:

$$e_{ji} = s_j^\top W^{(a)}[b_i^k, b_i^{K+1}],$$
$$c_j = \sum_{i=1}^{m} \alpha_{ji}[b_i^k, b_i^{K+1}].$$

In this model, the number of encoding parameters grows by a factor of $\frac{1}{k}$, and the number of decoding parameters grows by less than a factor of 2.

3.3 One-to-many model

Here, a single encoder is shared, while we keep a separate decoder for each domain. The shared encoder captures the fact that the input in each domain is a sequence of English words. The domain-specific decoders learn to output tokens from the right domain vocabulary.

4 Experiments

4.1 Data

We evaluated our system on the OVERNIGHT semantic parsing dataset, which contains 13,682 examples of language utterances paired with logical forms across eight domains. OVERNIGHT was constructed by generating logical forms from a grammar and annotating them with language through crowdsourcing. We evaluated on the same train/test split as Wang et al. (2015), using the same accuracy metric, that is, the proportion of questions for which the denotations of the predicted and gold logical forms are equal.

4.2 Implementation Details

We replicate the experimental setup of Jia and Liang (2016): We used the same hyper-parameters without tuning; we used 200 hidden units and 100-dimensional word vectors; we initialized parameters uniformly within the interval $[-0.1, 0.1]$, and maximized the log likelihood of the correct logical form with stochastic gradient descent. We trained the model for 30 epochs with an initial learning rate of 0.1, and halved the learning rate every 5 epochs, starting from epoch 15. We replaced word vectors for words that occur only once in the training set with a universal `<unk>` word vector. At test time, we used beam search with beam size 5. We then picked the highest-scoring logical form that does not yield an executor error when its denotation is computed. Our models were implemented in Theano (Bergstra et al., 2010).

Model	Basketball	Blocks	Calendar	Housing	Publications	Recipes	Restaurants	Social	Avg.	# Model Params
INDEP	85.2	61.2	77.4	67.7	74.5	79.2	79.5	80.2	75.6	14.1 M
MANY2MANY	83.9	63.2	79.8	75.1	75.2	81.5	79.8	**82.4**	77.6	22.8 M
ONE2MANY	84.4	59.1	79.8	74.6	80.1	81.5	80.7	81.1	77.7	8.6 M
INPUTTOKEN	85.9	63.2	79.2	77.8	75.8	80.6	**82.5**	81.0	78.2	2 M
ONE2ONE	84.9	**63.4**	75.6	76.7	78.9	**83.8**	81.3	81.4	78.3	2 M
DOMAINENCODING	**86.2**	62.7	**82.1**	**78.3**	**80.7**	82.9	82.2	81.7	**79.6**	2 M

Table 1: Test accuracy for all models on all domains, along with the number of parameters for each model.

4.3 Results

For our main result, we trained on all eight domains all models described in Section 3: ONE2ONE, DOMAINENCODING and INPUTTOKEN representing respectively the basic one-to-one model, with extensions of one-hot domain encoding or an extra input token, as described in Section 3.1. MANY2MANY and ONE2MANY are the models described in Sections 3.2 and 3.3, respectively. INDEP is the baseline sequence-to-sequence model described in Section 2, which trained independently on each domain.

Results show (Table 1) that training on multiple KBs improves average accuracy over all domains for all our proposed models, and that performance improves as more parameters are shared. Our strongest results come when parameter sharing is maximal (i.e., single encoder and single decoder), coupled with a one-hot domain representation at decoding time (DOMAINENCODING). In this case accuracy improves not only on average, but also for each domain separately. Moreover, the number of model parameters necessary for training the model is reduced by a factor of 7.

Our baseline, INDEP, is a reimplementation of the NORECOMBINATION model described in Jia and Liang (2016), which achieved average accuracy of 75.8% (corresponds to our 75.6% result). Jia and Liang (2016) also introduced a framework for generating new training examples in a single domain through *recombination*. Their model that uses the most training data achieved state-of-the-art average accuracy of 77.5% on OVERNIGHT. We show that by training over multiple KBs we can achieve higher average accuracy, and our best model, DOMAINENCODING, sets a new state-of-the-art average accuracy of 79.6%.

Figure 3 shows a learning curve for all models on the test set, when training on a fraction of the training data. We observe that the difference between models that share parameters (INPUTTOKEN, ONE2ONE and DOMAINENCODING) and models that keep most of the parameters separate (INDEP, MANY2MANY and ONE2MANY) is especially pronounced when the amount of data is small, reaching a difference of more than 15 accuracy point with 10% of the training data. This highlights the importance of using additional data from a similar distribution without increasing the number of parameters when there is little data. The learning curve also suggests that the MANY2MANY model improves considerably as the amount of data increases, and it would be interesting to examine its performance on larger datasets.

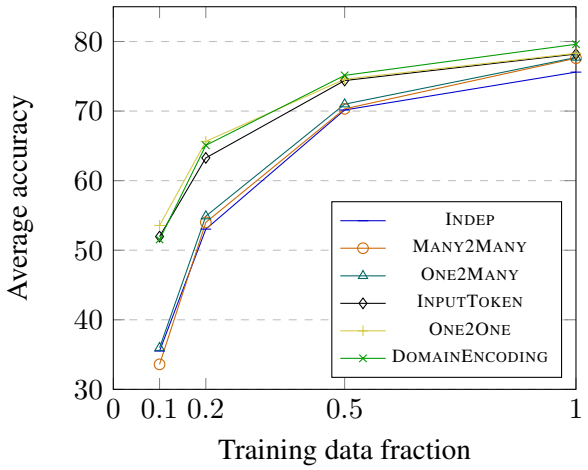

Figure 3: Learning curves for all models on the test set.

4.4 Analysis

Learning a semantic parser involves mapping language phrases to KB constants, as well as learning how language composition corresponds to logical form composition. We hypothesized that the main benefit of training on multiple KBs lies in learning about compositionality. To verify that, we append the domain index to the name of every constant in every KB, and therefore constant names are disjoint across datasets. We train DOMAINENCODING on this dataset and obtain an accuracy of 79.1% (comparing to 79.6%), which hints that most of the gain is attributed to compositionality rather than mapping of language to KB constants.

We also inspected cases where DOMAINEN-

CODING performed better than INDEP, by analyzing errors on a development set (20% of the training data). We found 45 cases where INDEP makes an error (and DOMAINENCODING does not) by predicting a wrong comparative or superlative structure (e.g., $>$ instead of $\geq$). However, the opposite case occurs only 29 times. This reiterates how we learn structural linguistic regularities when sharing parameters.

Lastly, we observed that the domain's training set size negatively correlates with its relative improvement in performance (DOMAINENCODING accuracy compared to INDEP), where Spearman's $\rho = -0.86$. This could be explained by the tendency of smaller domains to cover a smaller fraction of structural regularities in language, thus, they gain more by sharing information.

5 Conclusion

In this paper we address the challenge of obtaining training data for semantic parsing from a new perspective. We propose that one can improve parsing accuracy by training models over multiple KBs and demonstrate this on the eight domains of the OVERNIGHT dataset.

In future work, we would like to further reduce the burden of data gathering by training character-level models that learn to map language phrases to KB constants across datasets, and by pre-training language side models that improve the encoder from data that is independent of the KB. We also plan to apply this method on datasets where only denotations are provided rather than logical forms.

Reproducibility

All code, data, and experiments for this paper are available on the CodaLab platform at https://worksheets.codalab.org/worksheets/0xdec998f58deb4829aba80fbf49f69236/.

Acknowledgments

We thank Shimi Salant and the anonymous reviewers for their constructive feedback. This work was partially supported by the Israel Science Foundation, grant 942/16.

References

Y. Artzi and L. Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics (TACL)* 1:49–62.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

J. Berant and P. Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics (TACL)* 3:545–558.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Python for Scientific Computing Conference*.

J. Blitzer, S. Kakade, and D. P. Foster. 2011. Domain adaptation with coupled subspaces. In *Artificial Intelligence and Statistics (AISTATS)*. pages 173–181.

R. Caruana. 1997. Multitask learning. *Machine Learning* 28:41–75.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*. pages 18–27.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)* 12:2493–2537.

H. Daume III. 2007. Frustratingly easy domain adaptation. In *Association for Computational Linguistics (ACL)*.

L. Dong and M. Lapata. 2016. Language to logical form with neural attention. In *Association for Computational Linguistics (ACL)*.

O. Firat, K. Cho, and Y. Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *North American Association for Computational Linguistics (NAACL)*.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Vigas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558* .

J. Krishnamurthy and T. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*. pages 754–765.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1512–1523.

P. Liang. 2013. Lambda dependency-based compositional semantics. *arXiv* .

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*. pages 590–599.

M. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser. 2016. Multi-task sequence to sequence learning. In *International Conference on Learning Representations (ICLR)*.

M. Luong, H. Pham, and C. D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1412–1421.

S. Reddy, M. Lapata, and M. Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics (TACL)* 2(10):377–392.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. pages 3104–3112.

Y. Wang, J. Berant, and P. Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*. pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*. pages 658–666.