Language to Logical Form with Neural Attention

Li Dong and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
li.dong@ed.ac.uk mlap@inf.ed.ac.uk

Abstract

Semantic parsing aims at mapping natural language to machine interpretable meaning representations. Traditional approaches rely on high-quality lexicons, manually-built templates, and linguistic features which are either domain- or representation-specific. In this paper, we present a general method based on an attention-enhanced sequence-to-sequence model. We encode input sentences into vector representations using recurrent neural networks, and generate their logical forms by conditioning the output on the encoding vectors. The model is trained in an end-to-end fashion to maximize the likelihood of target logical forms given the natural language inputs. Experimental results on four datasets show that our approach performs competitively without using hand-engineered features and is easy to adapt across domains and meaning representations.

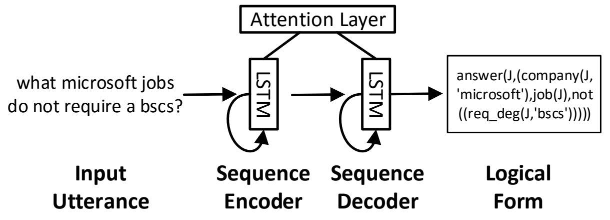

Attention Layer
what microsoft jobs do not require a bscs?
LSTM
LSTM
answer(J,(company(J,'microsoft'),job(J),not((req_deg(J,'bscs')))))
Input Utterance
Sequence Encoder
Sequence Decoder
Logical Form

Figure 1: We treat both input utterances and their logical forms as sequences which are encoded and decoded with different recurrent neural networks. The attention layer is used to learn soft alignments between natural language and logical form.

1 Introduction

Semantic parsing is the task of translating text to a formal meaning representation such as logical forms or structured queries. There has recently been a surge of interest in developing machine learning methods for semantic parsing (see the references in Section 2), due in part to the existence of corpora containing utterances annotated with formal meaning representations. Figure 1 shows an example of a question (left hand-side) and its annotated logical form (right hand-side), taken from JOBS (Tang and Mooney, 2001), a well-known semantic parsing benchmark. In order to predict the correct logical form for a given input utterance, most previous systems rely on predefined templates and manually designed features, which often render parsers domain- or representation-specific. In this work, we aim to use a simple yet effective method to bridge the gap between natural language and logical form with minimal domain knowledge.

Sequence transduction architectures based on recurrent neural networks have been successfully applied to a variety of NLP tasks ranging from syntactic parsing (Vinyals et al., 2015a), to machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014), and image description generation (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015b). An attention mechanism is often used to improve performance for long sequences (Bahdanau et al., 2015; Vinyals et al., 2015a). We adapt the general sequence-to-sequence modeling paradigm to the semantic parsing task. As shown in Figure 1, our model learns from natural language descriptions paired with meaning representations; it encodes sentences and decodes logical forms using recurrent neural networks with long short-term memory (LSTM) units. We also introduce an attention mechanism (Bahdanau et al., 2015; Luong et al., 2015b) which allows the model to learn soft alignments between natural language and logical forms. Rare words present a problem to sequence transduction models, and semantic parsing is no exception. In order to handle rare words, we mask entities and numbers with their types and recover them in a post-processing step. Evaluation results demonstrate that our model achieves similar or better performance when compared to previous methods across different domains and meaning representations, despite using no hand-engineered features, templates, domain or linguistic knowledge.

The contributions of our work are three-fold: (1) we present a general-purpose sequence-to-sequence model for semantic parsing; (2) propose a method to address the rare word problem for our neural model; and (3) conduct extensive experiments on several benchmark datasets providing detailed analysis on what our model learns.

2 Related Work

Our work synthesizes two strands of research, namely semantic parsing and the encoder-decoder architecture with neural networks.

The problem of learning semantic parsers has received significant attention, dating back to Woods (1973). A large number of approaches learn from sentences paired with logical forms following different modeling strategies. Examples include the use of parsing models (Miller et al., 1996; Ge and Mooney, 2005; Lu et al., 2008; Zhao and Huang, 2015), inductive logic programming (Zelle and Mooney, 1996; Tang and Mooney, 2000; Thomspon and Mooney, 2003), probabilistic automata (He and Young, 2006), string/tree-to-tree transformation rules (Kate et al., 2005), classifiers based on string kernels (Kate and Mooney, 2006), machine translation (Wong and Mooney, 2006; Wong and Mooney, 2007; Andreas et al., 2013), and combinatory categorial grammar induction techniques (Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2010; Kwiatkowski et al., 2011). Other work learns semantic parsers without relying on logical-from annotations, e.g., from sentences paired with conversational logs (Artzi and Zettlemoyer, 2011), system demonstrations (Chen and Mooney, 2011; Goldwasser and Roth, 2011; Artzi and Zettlemoyer, 2013), question-answer pairs (Clarke et al., 2010; Liang et al., 2013), and distant supervision (Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013; Reddy et al., 2014).

Our model learns from natural language descriptions paired with meaning representations. Most previous systems rely on high-quality lexicons, manually-built templates, and features which are either domain- or representation-specific. We instead present a general method that can be easily adapted to different domains and meaning representations. We adopt the general encoder-decoder framework based on neural networks which has been recently repurposed for various NLP tasks such as syntactic parsing (Vinyals et al., 2015a), machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014), image description generation (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015b), question answering (Hermann et al., 2015), and summarization (Rush et al., 2015).

Mei et al. (2016) use a sequence-to-sequence model to map navigational instructions to actions. Our model works on more well-defined meaning representations (such as Prolog and lambda calculus) and is conceptually simpler; it does not employ bidirectionality or multi-level alignments. Grefenstette et al. (2014) propose a different architecture for semantic parsing based on the combination of two neural network models. The first model learns shared representations from pairs of questions and their translations into knowledge base queries, whereas the second model generates the queries conditioned on the learned representations. However, they do not report empirical evaluation results.

3 Model

Our goal is to learn a model which maps natural language input $q = \left(x_1, \cdots, x_{|q|}\right)$ to a logical form representation of its meaning $a = \left(y_1, \cdots, y_{|a|}\right)$. We regard both input q and output a as sequences and wish to estimate the conditional probability $p\left(a|q\right)$ which is decomposed as follows:

$$p\left(a|q\right) = \prod_{t=1}^{|a|} p\left(y_t|y_{<t}, q\right) \tag{1}$$

where $y_{<t} = (y_1, \cdots, y_{t-1})$. In the following we describe the details of how $p\left(a|q\right)$ is computed.

3.1 Sequence-to-Sequence Model

Our model consists of an **encoder** which encodes natural language input q into a vector representation and a **decoder** which learns to generate $y_1, \cdots, y_{|a|}$ conditioned on the encoding vector.

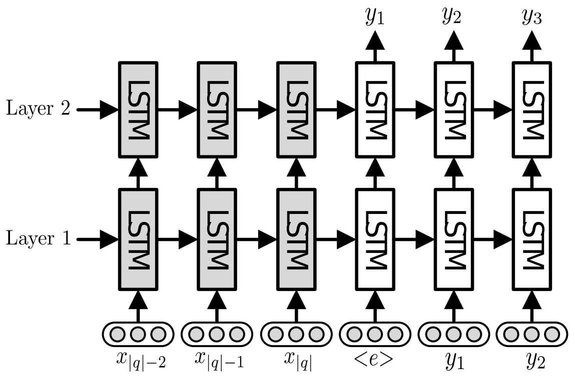


Figure 2: A sequence-to-sequence model with two-layer recurrent neural networks.

As shown in Figure 2, the encoder and decoder are two different L-layer recurrent neural networks with Long Short-Term Memory (LSTM) units which recursively process tokens one by one. The first $|q|$ time steps belong to the encoder, while the following $|a|$ time steps belong to the decoder. Let $\mathbf{h}_t^l \in \mathbb{R}^n$ denote an n-dimensional hidden vector in time step t and layer l. $\mathbf{h}_t^l$ is computed by:

$$\mathbf{h}_t^l = f\left(\mathbf{h}_{t-1}^l, \mathbf{h}_t^{l-1}\right) \tag{2}$$

where $\mathbf{h}_t^0$ is the word vector of the current token for the encoder or the previously predicted word for the decoder. Specifically, we follow the definition of the LSTM unit used in Zaremba et al. (2015) to recursively compute the hidden vectors:

$$\begin{aligned}\begin{pmatrix}\mathbf{i}\\ \mathbf{f}\\ \mathbf{o}\\ \mathbf{g}\end{pmatrix} &= \begin{pmatrix}\text{sigm}\\ \text{sigm}\\ \text{sigm}\\ \tanh\end{pmatrix} W^l \begin{pmatrix}\mathbf{h}_t^{l-1}\\ \mathbf{h}_{t-1}^l\end{pmatrix}\\ \mathbf{m}_t^l &= \mathbf{f} \odot \mathbf{m}_{t-1}^l + \mathbf{i} \odot \mathbf{g}\\ \mathbf{h}_t^l &= \mathbf{o} \odot \tanh\left(\mathbf{m}_t^l\right)\end{aligned} \tag{3}$$

where $\tanh$ and sigm are applied element-wise, $\odot$ is element-wise multiplication, and $W^l \in \mathbb{R}^{4n \times 2n}$ is a parameter matrix for the l-th layer. Notice that the encoder and decoder have different parameters.

Once the tokens of the input sequence $x_1, \cdots, x_{|q|}$ are encoded into vectors, they are used to initialize the hidden states of the first time step in the decoder. Next, the hidden vector of the topmost LSTM $\mathbf{h}_{|q|+t}^L$ is used to predict the t-th output token as:

$$p\left(y_t | y_{<t}, q\right) = \text{softmax}\left(W_p \mathbf{h}_{|q|+t}^L\right)^\intercal \mathbf{e}\left(y_t\right) \tag{4}$$

where $W_p \in \mathbb{R}^{|V_a| \times n}$ is a parameter matrix, and $\mathbf{e}\left(y_t\right) \in \{0,1\}^{|V_a|}$ is a one-hot vector used to obtain y_t's probability from the predicted distribution.

We augment every sequence with a "start-of-sequence" $<s>$ and "end-of-sequence" $<e>$ token. The generation process terminates once $<e>$ is predicted. Then, the conditional probability of generating the whole sequence $p\left(a|q\right)$ is computed using Equation (1).

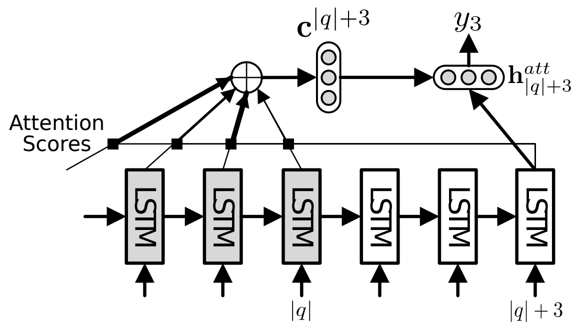


Figure 3: Attention scores are computed using the current hidden vector of the decoder and all hidden vectors of the encoder obtained in the form of a weighted sum. The new hidden vector $\mathbf{h}_{|q|+3}^{att}$ is used to predict y_3 for the current time step.

3.2 Attention Mechanism

As shown in Equation (4), the hidden vectors of the input sequence are not directly used in the decoding process. However, it makes intuitively sense to consider relevant information from the input to better predict the current token. Following this idea, various techniques have been proposed to integrate encoder-side information (in the form of a context vector) into each time step of the decoder (Bahdanau et al., 2015; Luong et al., 2015b; Xu et al., 2015; Hermann et al., 2015). We use the global attention architecture described in Luong et al. (2015b).

As shown in Figure 3, in order to find relevant encoder-side context for the current hidden state $\mathbf{h}_{|q|+t}^L$, we compute its attention score with the k-th hidden state in the encoder as:

$$s_k^{|q|+t} = \frac{\exp\{(\mathbf{h}_k^L)^\intercal \mathbf{h}_{|q|+t}^L\}}{\sum_{j=1}^{|q|} \exp\{(\mathbf{h}_j^L)^\intercal \mathbf{h}_{|q|+t}^L\}} \tag{5}$$

where only the top-layer hidden vectors are utilized. Then, the context vector is the weighted sum of the hidden vectors in the encoder:

$$\mathbf{c}^{|q|+t} = \sum_{k=1}^{|q|} s_k^{|q|+t} \mathbf{h}_k^L \tag{6}$$

In lieu of Equation (4), we further use this context vector which acts as a summary of the encoder to compute the probability of generating y_t as:

$$\mathbf{h}_{|q|+t}^{att} = \tanh\left(W_1 \mathbf{h}_{|q|+t}^L + W_2 \mathbf{c}^{|q|+t}\right) \tag{7}$$

$$p\left(y_t | y_{<t}, q\right) = \text{softmax}\left(W_p \mathbf{h}_{|q|+t}^{att}\right)^\intercal \mathbf{e}\left(y_t\right) \tag{8}$$

where $W_p \in \mathbb{R}^{|V_a| \times n}$ and $W_1, W_2 \in \mathbb{R}^{n \times n}$ are three parameter matrices, and $\mathbf{e}\left(y_t\right)$ is a one-hot vector used to obtain y_t's probability.

3.3 Model Training

Our goal is to maximize the likelihood of generated logical forms given natural language utterances as input. So the objective function is defined as:

$$\min - \sum_{(q,a) \in \mathcal{D}} \log p\left(a|q\right) \tag{9}$$

where $\mathcal{D}$ is the set of all natural language-logical form training pairs, and $p\left(a|q\right)$ is computed as shown in Equation (1). The RMSProp algorithm (Tieleman and Hinton, 2012) is employed to solve this non-convex optimization problem. Moreover, dropout is used for regularizing the model as suggested in Zaremba et al. (2015). Specifically, dropout operators are used between different LSTM layers and for the hidden layers before the softmax classifiers. This technique can substantially reduce overfitting, especially on datasets of small size.

3.4 Inference

At test time, the model predicts the logical form for an input utterance q maximizing the conditional probability:

$$\hat{a} = \arg\max_{a'} p\left(a'|q\right) \tag{10}$$

where a' represents a candidate logical form sequence. However, it is not practical to iterate over all possible sequences to obtain the optimal prediction. According to Equation (1), we decompose the probability $p\left(a|q\right)$ so that we can use beam search or greedy search to predict tokens one by one. We add special tokens $<s>$ and $<e>$ to the beginning and end of every sequence. The generation process terminates once token $<e>$ is predicted.

3.5 Handling Rare Words

Some entities and numbers are rare or even do not appear in the training set at all, especially on small-scale datasets. Conventional sequence-to-sequence models replace rare words with the unknown word symbol (Luong et al., 2015a; Jean et al., 2015). This strategy, however, would adversely affect our model's accuracy, because a prediction result is correct only if all tokens are parsed correctly.

Our solution is to link entities and numbers in utterances to their corresponding logical constants, and replace them with their type names and unique IDs. For instance, we pre-process the training example "*jobs with a salary of 40000*" and its logical form "job(ANS), salary_greater_than(ANS, 40000, year)" as "*jobs with a salary of* $\underline{\mathit{num}_0}$" and "job(ANS), salary_greater_than(ANS, $\underline{\mathit{num}_0}$, year)". We use the pre-processed examples as training data. At inference time, we also mask entities and numbers with their types and IDs. Once we obtain the decoding result, a post-processing step recovers all the markers $\underline{\mathtt{type}_i}$ to their corresponding logical constants.

4 Experiments

We compare our method against multiple previous systems on four datasets. We describe these datasets below, and present our experimental settings and results. Finally, we conduct model analysis in order to understand what the model learns.

4.1 Datasets

Our model was trained on the following datasets, covering different domains and using different meaning representations. Examples for each domain are shown in Table 1.

JOBS This benchmark dataset contains 640 queries to a database of job listings. Specifically, questions are paired with Prolog-style queries. We used the same training-test split as Zettlemoyer

Dataset	Length	Example
JOBS	9.80	*what microsoft jobs do not require a bscs?*
	22.90	answer(company(J,'microsoft'),job(J),not((req_deg(J,'bscs'))))
GEO	7.60	*what is the population of the state with the largest area?*
	19.10	(population:i (argmax \$0 (state:t \$0) (area:i \$0)))
ATIS	11.10	*dallas to san francisco leaving after 4 in the afternoon please*
	28.10	(lambda \$0 e (and (>(departure_time \$0) 1600:ti) (from \$0 dallas:ci) (to \$0 san_francisco:ci)))
IFTTT	6.95	*Turn on heater when temperature drops below 58 degree*
	21.80	TRIGGER: Weather - Current_temperature_drops_below - ((Temperature (58)) (Degrees_in (f)))
		ACTION: WeMo_Insight_Switch - Turn_on - ((Which_switch? ("")))

Table 1: Examples of natural language descriptions and their meaning representations from four datasets. The average length of input and output sequences is shown in the second column.

and Collins (2005) which contains 500 training and 140 test instances. Values for the variables company, degree, language, platform, location, job area, and number are considered as rare words.

GEO This is a standard semantic parsing benchmark which contains 880 queries to a database of U.S. geography. GEO has 880 sentence logical-form pairs split into a training set of 680 training examples and 200 test examples (Zettlemoyer and Collins, 2005). We used the same meaning representation based on lambda-calculus as Kwiatkowski et al. (2011). Values for the varibles city, state, country, river, and number are handled as rare words.

ATIS This dataset has $5,410$ queries to a flight booking system. The standard split (Kwiatkowski et al., 2011) has $4,480$ training instances, 480 development instances, and 450 test instances. Sentences are paired with lambda-calculus expressions. Values for the variables date, time, city, aircraft code, airport, airline, and number are treated as rare words.

IFTTT Quirk et al. (2015) created this dataset by extracting a large number of if-this-then-that recipes from the IFTTT website[1]. Recipes are simple programs with exactly one trigger and one action which users specify on the site. Whenever the conditions of the trigger are satisfied, the action is performed. Actions typically revolve around home security (e.g., "*turn on my lights when I arrive home*"), automation (e.g., "*text me if the door opens*"), well-being (e.g., "*remind me to drink water if I've been at a bar for more than two hours*"), and so on. Triggers and actions are selected from different channels (160 in total) representing various types of services, devices (e.g., Android), and knowledge sources (such as ESPN or Gmail). In the dataset, there are 552 trigger functions from 128 channels, and 229 action functions from 99 channels. We used Quirk et al.'s (2015) original split which contains $77,495$ training, $5,171$ development, and $4,294$ test examples. The IFTTT programs are represented as abstract syntax trees and are paired with natural language descriptions provided by users (see Table 1). Here, numbers and URLs are handled as rare words.

[1]`http://www.ifttt.com`

4.2 Settings

We converted natural language sentences to lowercase and corrected misspelled words using a dictionary based on the Wikipedia list of common misspellings. To reduce sparsity, words were stemmed using the Snowball Stemmer within NLTK (Bird et al., 2009). For IFTTT, we filtered tokens, channels and functions which appeared less than five times in the training set. For the other datasets, we filtered input words which did not occur at least two times in the training set, but kept all tokens in the logical forms. Plain string matching was employed to identify entities for handling rare words as described in Section 3.5. More sophisticated approaches could be used, however we leave this future work.

Model hyper-parameters were cross-validated on the training set for JOBS and GEO. We used the standard development sets for ATIS and IFTTT. Our model was trained with mini-batch stochastic gradient descent using the RMSProp algorithm (with batch size set to 20). The smoothing constant of RMSProp was set to 0.95. The base learning rate was 0.002 for IFTTT and 0.01 for the other do-

Method	Accuracy
COCKTAIL (Tang and Mooney, 2001)	79.4
PRECISE (Popescu et al., 2003)	88.0
ZC05 (Zettlemoyer and Collins, 2005)	79.3
DCS+L (Liang et al., 2013)	90.7
TISP (Zhao and Huang, 2015)	85.0
seq2seq − rare	70.7
seq2seq − attention	77.9
seq2seq (ours)	87.1

Table 2: Evaluation results on JOBS.

mains. After 5 epochs, the learning rate was decreased by a factor of 0.95 at every epoch as suggested in Karpathy et al. (2015). Gradients were clipped at 5 to alleviate the exploding gradient problem (Pascanu et al., 2013). Parameters were initialized by randomly sampling values from a uniform distribution $\mathcal{U}(-0.08, 0.08)$. A two-layer LSTM was used for IFTTT, while a one-layer LSTM was employed for the other domains. We also used dropout regularization; the dropout rate was selected from $\{0.2, 0.3, 0.4, 0.5\}$. Dimensions of the hidden vector and the word embedding were the same, and selected from $\{150, 200, 250\}$. Early stopping was employed to determine the number of epochs. The input sentences were reversed before feeding into the sequence encoder as suggested by Sutskever et al. (2014). Greedy search was used to generate logical forms during inference. Notice that two decoders with shared word embeddings were used to predict triggers and actions for IFTTT. All experiments were conducted on a single GTX 980 GPU device.

4.3 Results

We first discuss the performance of our model on JOBS, GEO, and ATIS, and then examine our results on IFTTT. Tables 2–4 compare our results against a variety of systems previously described in the literature. In addition to our model (seq2seq), we present two ablation variants, namely without using an attention mechanism ($-$attention) and without handing rare words ($-$rare). We report accuracy which is defined as the proportion of input sentences that are correctly parsed to their gold standard logical forms. Notice that DCS+L, KCAZ13 and GUSP output answers directly, so accuracy in this setting is defined as the percentage of correct answers.

We find that adding attention substantially improves performance on all three datasets. This underlines the importance of utilizing soft alignments between inputs and outputs. We further analyze what the attention layer learns in Section 4.4. Moreover, the results show that handling rare words is critical for small-scale datasets. For example, about 92% of city names appear less than 4 times in the GEO training set, so it is difficult to learn reliable parameters for these words. In relation to previous models, the seq2seq framework achieves comparable or better performance without relying on manually defined templates or language/domain specific features. Importantly, we use the same model across datasets and meaning representations (Prolog-style logical forms in JOBS and lambda calculus in the

Method	Accuracy
SCISSOR (Ge and Mooney, 2005)	72.3
SILT (Kate et al., 2005)	54.1
KRISP (Kate and Mooney, 2006)	71.7
WASP (Wong and Mooney, 2006)	74.8
λ-WASP (Wong and Mooney, 2007)	86.6
LNLZ08 (Lu et al., 2008)	81.8
ZC05 (Zettlemoyer and Collins, 2005)	79.3
ZC07 (Zettlemoyer and Collins, 2007)	86.1
UBL (Kwiatkowski et al., 2010)	87.9
FUBL (Kwiatkowski et al., 2011)	88.6
KCAZ13 (Kwiatkowski et al., 2013)	89.0
DCS+L (Liang et al., 2013)	87.9
TISP (Zhao and Huang, 2015)	88.9
seq2seq − rare	68.6
seq2seq − attention	72.9
seq2seq (ours)	84.6

Table 3: Evaluation results on GEO. 10-fold cross-validation is used for the systems shown in the top half of the table. The standard split of ZC05 is used for all other systems.

Method	Accuracy
ZC07 (Zettlemoyer and Collins, 2007)	84.6
UBL (Kwiatkowski et al., 2010)	71.4
FUBL (Kwiatkowski et al., 2011)	82.8
GUSP-FULL (Poon, 2013)	74.8
GUSP++ (Poon, 2013)	83.5
TISP (Zhao and Huang, 2015)	84.2
seq2seq − rare	74.1
seq2seq − attention	78.5
seq2seq (ours)	84.2

Table 4: Evaluation results on ATIS.

Method	Channel	+Func	F1
retrieval	28.9	20.2	41.7
phrasal	19.3	11.3	35.3
sync	18.1	10.6	35.1
classifier	48.8	35.2	48.4
posclass	50.0	36.9	49.3
seq2seq − rare	53.9	38.6	49.7
seq2seq − attention	54.0	37.9	49.8
seq2seq (ours)	54.3	39.2	50.1

(a) Omit non-English.

Method	Channel	+Func	F1
retrieval	36.8	25.4	49.0
phrasal	27.8	16.4	39.9
sync	26.7	15.5	37.6
classifier	64.8	47.2	56.5
posclass	67.2	50.4	57.7
seq2seq − rare	68.8	50.4	59.7
seq2seq − attention	68.7	48.9	59.5
seq2seq (ours)	68.8	50.5	60.3

(b) Omit non-English & unintelligible.

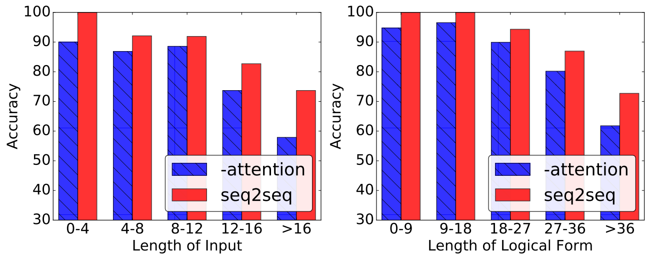
Method	Channel	+Func	F1
retrieval	43.3	32.3	56.2
phrasal	37.2	23.5	45.5
sync	36.5	24.1	42.8
classifier	79.3	66.2	65.0
posclass	81.4	71.0	66.5
seq2seq − rare	86.8	74.9	70.8
seq2seq − attention	88.3	73.8	72.9
seq2seq (ours)	87.8	75.2	73.7

(c) ≥ 3 turkers agree with gold.

Table 5: Evaluation results on IFTTT.

other two datasets), without modification. Despite this relatively simple approach, we observe that seq2seq ranks third on JOBS, and second on ATIS.

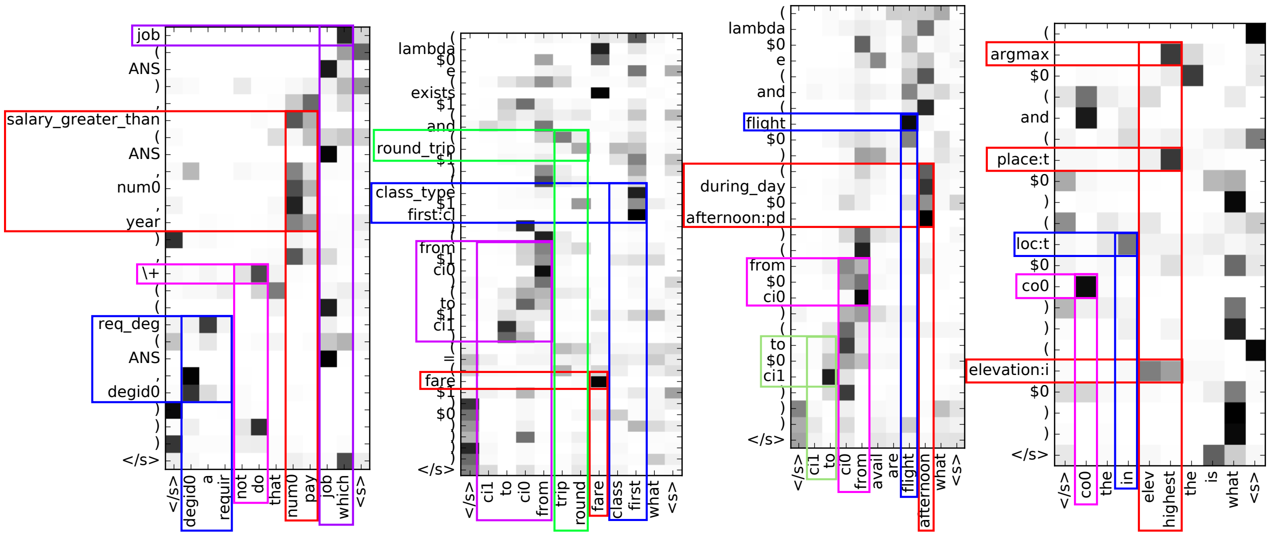
For IFTTT, we follow the same evaluation protocol introduced in Quirk et al. (2015). The dataset is extremely noisy and measuring accuracy is problematic since predicted abstract syntax trees (ASTs) almost never exactly match the gold standard. Quirk et al. view an AST as a set of productions and compute balanced F1 instead which we also adopt. The first column in Table 5 shows the percentage of channels selected correctly for both triggers and actions. The second column measures accuracy for *both* channels and functions. The last column shows balanced F1-measure against the gold tree over all productions in the proposed derivation. We compare our model against posclass, the method introduced in Quirk et al. and several of their baselines. posclass is reminiscent of KRISP (Kate and Mooney, 2006), it learns distributions over productions given input sentences represented as a bag of linguistic features. The retrieval baseline searches for the closest description in the training data based on character string-edit-distance and returns the recipe for that training program. The phrasal method uses phrase-based machine translation to generate the recipe, whereas sync extracts synchronous grammar rules from the data, essentially recreating WASP (Wong and Mooney, 2006). Finally, the classifier approach uses a binary classifier to predict whether a production should be present in the derivation tree corresponding to the description.



(a) (b)

Figure 5: Accuracy as a function of (a) input and (b) output length (ATIS, development set).

Quirk et al. (2015) report results on the full test data and smaller subsets after noise filtering, e.g., when non-English and unintelligible descriptions are removed (Tables 5a and 5b). They also ran their system on a high-quality subset of description-program pairs which were found in the gold standard and at least three humans managed to independently reproduce (Table 5c). As can be seen, across all subsets seq2seq outperforms posclass and related baselines. Compared to the previous datasets, the attention mechanism and the method used to handle rare words yield less of an improvement here. This may be due to the size of Quirk et al. (2015) and the way it was created – user curated descriptions are often of low quality, and thus align very loosely to their corresponding ASTs.



(a) which jobs pay num0 that do not require a degid0

(b) what's first class fare round trip from ci0 to ci1

(c) what afternoon flights are available from ci0 to ci1

(d) what is the highest elevation in the co0

Figure 4: Examples of alignments (same color rectangles) produced by the model's attention mechanism (darker color represents higher attention score). Input sentences are reversed and stemmed.

4.4 Model Analysis

Figure 4 illustrates examples of alignments produced by our model. Matrices of attention scores are computed using Equation (5) and are represented in grayscale. Aligned input words and logical form predicates are enclosed in (same color) rectangles whose overlapping areas contain the attention scores. Input sentences are reversed (Sutskever et al., 2014) for better performance. Also notice that attention scores are computed by LSTM hidden vectors which encode context information rather than just the words in their current positions. The examples demonstrate that the attention mechanism learns soft alignments between input sentences and output logical forms, performing surprisingly well, even in cases of reordering (Figure 4b, 4c and 4d) and one-to-many alignments (Figure 4c).

We also explore whether the length of natural language descriptions and logical forms influences performance. Figure 5 shows how accuracy varies with different length ranges on the ATIS development set. Overall, we observe that accuracy drops as sequence length becomes larger, because longer sequences tend to have more complex meanings and be more compositional. We also find that the attention mechanism consistently boosts performance across all ranges of length. Moreover, adding attention allows the system to better cope with longer sequences compared to the vanilla seq2seq method.

5 Conclusions

In this paper we presented a sequence-to-sequence model for semantic parsing. Given natural language descriptions as input our model predicts meaning representations as output. Both input sentences and output logical forms are regarded as sequences, and are encoded and decoded by different recurrent neural networks. Our experimental results show that enhancing the model with an attention mechanism improves performance across the board, especially for long sequences. We also show that our method of handling rare words (via masking and post-processing) is effective and boosts performance. Extensive comparisons with previous approaches demonstrate that our model performs competitively, without recourse to domain- or representation-specific features. Directions for future work are many and varied. For example, it would be interesting to learn a model from question-answer pairs without access to target logical forms. We would also like to use the model for question answering over a knowledge base, e.g., by ranking the predicate paths between entities (figuring in the queries) and candidate answers.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *ACL*, pages 47–52, Sofia, Bulgaria.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 EMNLP*, pages 421–432, Edinburgh, Scotland, UK.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*, 1(1):49–62.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the ICLR*, San Diego, CA.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Qingqing Cai and Alexander Yates. 2013. Semantic parsing freebase: Towards open-domain semantic parsing. In *2nd Joint Conference on Lexical and Computational Semantics*, pages 328–338, Atlanta, GA.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 15th AAAI*, pages 859–865, San Francisco, CA.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 EMNLP*, pages 1724–1734, Doha, Qatar.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of CONLL*, pages 18–27, Uppsala, Sweden.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of CoNLL*, pages 9–16, Ann Arbor, MI.

Dan Goldwasser and Dan Roth. 2011. Learning from natural instructions. In *Proceedings of the 22nd IJCAI*, pages 1794–1800, Barcelona, Spain.

Edward Grefenstette, Phil Blunsom, Nando de Freitas, and Karl Moritz Hermann. 2014. A deep architecture for semantic parsing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, Atlanta, GA.

Yulan He and Steve Young. 2006. Semantic processing using the hidden vector state model. *Speech Communication*, 48(3-4):262–275.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*, pages 1684–1692. Curran Associates, Inc.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of 53rd ACL and 7th IJCNLP*, pages 1–10, Beijing, China.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 EMNLP*, pages 1700–1709, Seattle, WA.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of CVPR*, pages 3128–3137, Boston, MA.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st COLING and 44th ACL*, pages 913–920, Sydney, Australia.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th AAAI*, pages 1062–1068, Pittsburgh, PA.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 EMNLP*, pages 754–765, Jeju Island, Korea.

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 EMNLP*, pages 1223–1233, Cambridge, MA.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 EMNLP*, pages 1512–1523, Edinburgh, United Kingdom.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 EMNLP*, pages 1545–1556, Seattle, WA.

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

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceed-*

ings of the 2008 EMNLP, pages 783–792, Honolulu, Hawaii.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015a. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd ACL and 7th IJCNLP*, Beijing, China.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 EMNLP*, pages 1412–1421, Lisbon, Portugal.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the 30th AAAI*, Phoenix, AZ. to appear.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *ACL*, pages 55–61.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th ICML*, pages 1310–1318, Atlanta, GA.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st ACL*, pages 933–943, Sofia, Bulgaria.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 149–157, Miami, FL.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *ACL*, pages 878–888, Beijing, China, July.

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

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 EMNLP*, pages 379–389, Lisbon, Portugal.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *EMNLP*, pages 133–141, Hong Kong, China.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *ECML*, pages 466–477, Freiburg, Germany.

Cynthia A. Thomspon and Raymond J. Mooney. 2003. Acquiring word-meaning mappings for natural language interfaces. *JAIR*, 18:1–44.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. Technical report.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015a. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28*, pages 2755–2763.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *Proceedings of CVPR*, pages 3156–3164, Boston, MA.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of NAACL*, pages 439–446, New York, NY.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th ACL*, pages 960–967, Prague, Czech Republic.

W. A. Woods. 1973. Progress in natural language understanding: An application to lunar geology. In *Proceedings of the June 4-8, 1973, National Computer Conference and Exposition*, pages 441–450, New York, NY.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd ICML*, pages 2048–2057, Lille, France.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. In *Proceedings of the ICLR*, San Diego, CA.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the 19th AAAI*, pages 1050–1055, Portland, Oregon.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st UAI*, pages 658–666, Toronto, ON.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *In Proceedings of the EMNLP-CoNLL*, pages 678–687, Prague, Czech Republic.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of NAACL*, pages 1416–1421, Denver, CO.