Supplementary Materials for EMNLP 2018 Paper: Neural Compositional Denotational Semantics for Question Answering

Nitish Gupta[*]
University of Pennsylvania
Philadelphia, PA
nitishg@seas.upenn.edu

Mike Lewis
Facebook AI Research
Seattle, WA
mikelewis@fb.com

A Training Objective

Given a dataset $\mathcal{D}$ of (question, answer, knowledge-graph) tuples, $\{\mathrm{q}^i, \mathrm{a}^i, \mathrm{KG}^i\}_{i=1}^{i=|\mathcal{D}|}$, we train our model to maximize the log-likelihood of the correct answers. Answers are either booleans ($a \in \{0, 1\}$), or specific subsets of entities ($a = \{e_j\}$) from the KG. We denote the semantic type of the answer as a_t. The model's answer is found by taking the complete question representation, containing a distribution over types and the representation for each type. We maximize the following objective:

$$\mathcal{L} = \sum_i \log p(\mathrm{a}^i|\mathrm{q}^i, \mathrm{KG}^i) \tag{1}$$

$$= \sum_i \mathcal{L}_b^i + \mathcal{L}_e^i \tag{2}$$

where $\mathcal{L}_b^i$ and $\mathcal{L}_e^i$ are respectively the objective functions for questions with boolean answers and entity set answers.

$$\mathcal{L}_b^i = \mathbb{1}_{a_t^i=\mathbf{T}} \Big[\log(p_{true})^{a^i} (1 - p_{true})^{(1-a^i)} \Big] \tag{3}$$

$$\mathcal{L}_e^i = \frac{\mathbb{1}_{a_t^i=\mathbf{E}}}{|\mathcal{E}^i|} \Big[\log \prod_{e_j^i \in a^i} p_{e_j^i} \prod_{e_j^i \notin a^i} (1 - p_{e_j^i}) \Big] \tag{4}$$

We also add L_2-regularization for the scalar parsing features introduced in § **??**.

B Training Details

Representing Entities: Each entity in CLEVRGEN and GENX datasets consists of 4 attributes. For each attribute-value, we learn an embedding vector and concatenate these vectors to form the representation for the entity.

Training details: For curriculum learning, for the CLEVRGEN dataset we use a 2-step schedule where we first train our model on simple attribute match (*What is a red sphere?*), attribute existence (*Is anything blue?*) and boolean composition (*Is anything green and is anything purple?*) questions and in the second step on all questions. For GENX we use a 5-step, question-length based schedule, where we first train on shorter questions and eventually on all questions.

We tune hyper-parameters using validation accuracy on the CLEVRGEN dataset, and use the same hyper-parameters for both datasets. We train using SGD with a learning rate of 0.5, a mini-batch size of 4, and regularization constant of 0.3. When assigning the semantic type distribution to the words at the leaves, we add a small positive bias of $+1$ for $\boldsymbol{\phi}$-type and a small negative bias of -1 for the **E**-type score before the softmax. Our trainable parameters are: question word embeddings (64-dimensional), relation embeddings (64-dimensional), entity attribute-value embeddings (16-dimensional), four vectors per word for **V**-type representations, parameter vector θ for the parsing model that contains six scalar feature scores per module per word, and the global parameter vector for the **E+E→E** module.

C Baseline Models

C.1 LSTM (No KG)

We use a LSTM network to encode the question as a vector q. We also define three other parameter vectors, t, e and b that are used to predict the answer-type $P(a = \mathbf{T}) = \sigma(q \cdot t)$, entity attention value $p_{e_i} = \sigma(q \cdot e)$, and the probability of the answer being True $p_{true} = \sigma(q \cdot b)$.

[*] Work done while interning with Facebook AI Research.

C.2 LSTM

Similar to LSTM (NO RELATION), the question is encoded using a LSTM network as vector q. Similar to our model, we learn entity attribute-value embeddings and represent each entity as the concatenation of the 4 attribute-value embeddings, v_{e_i}. Similar to LSTM (NO RELATION), we also define the t parameter vector to predict the answer-type. The entity-attention values are predicted as $p_{e_i} = \sigma(v_{e_i} \cdot q)$. To predict the probability of the boolean-type answer being true, we first add the entity representations to form $b = \sum_{e_i} v_{e_i}$, then make the prediction as $p_{true} = \sigma(q \cdot b)$.

C.3 Tree-LSTM

Training the Tree-LSTM model requires pre-parsed sentences for which we use a binary constituency tree generating PCFG parser (?). We input the pre-parsed question to the Tree-LSTM to get the question embedding q. The rest of the model is same the LSTM model above.

C.4 Relation Network Augmented Model

The original formulation of the relation network module is as follows:

$$RN(q, KG) = f_\phi\left(\sum_{i,j} g_\theta(e_i, e_j, q)\right) \tag{5}$$

where e_i, e_j are the representations of the entities and q is the question representation from an LSTM network. The output of the Relation Network module is a scalar score value for the elements in the answer vocabulary. Since our dataset contains entity-set valued answers, we modified the module in the following manner.

We concatenate the entity-pair representations with the representations of the pair of relations between them[1]. We use the RN-module to produce an output representation for each entity as:

$$RN_{e_i} = f_\phi\left(\sum_{j} g_\theta(e_i, e_j, r^1_{ij}, r^2_{ij}, q)\right) \tag{6}$$

Similar to the LSTM baselines, we define a parameter vector t to predict the answer-type, and compute the vector b to compute the probability of the boolean type answer being true.

[1]In the CLEVR dataset, between any pair of entities, only 2 directed relations, *left* or *right*, and *above* or *beneath* are present.

To predict the entity-attention values, we use a separate attribute-embedding matrix to first generate the output representation for each entity, e_i^{out}, then predict the output attention values as follows:

$$p_{e_i} = \sigma\left(RN_{e_i} \cdot e_i^{out}\right) \tag{7}$$

We tried other architectures as well, but this modification provided the best performance on the validation set. We also tuned the hyper-parameters and found the setting from **?** to work the best based on validation accuracy. We used a different 2-step curriculum to train the RELATION NETWORK module, in which we replace the Boolean questions with the relation questions in the first-schedule and jointly train on all questions in the subsequent schedule.

C.5 SEMPRE

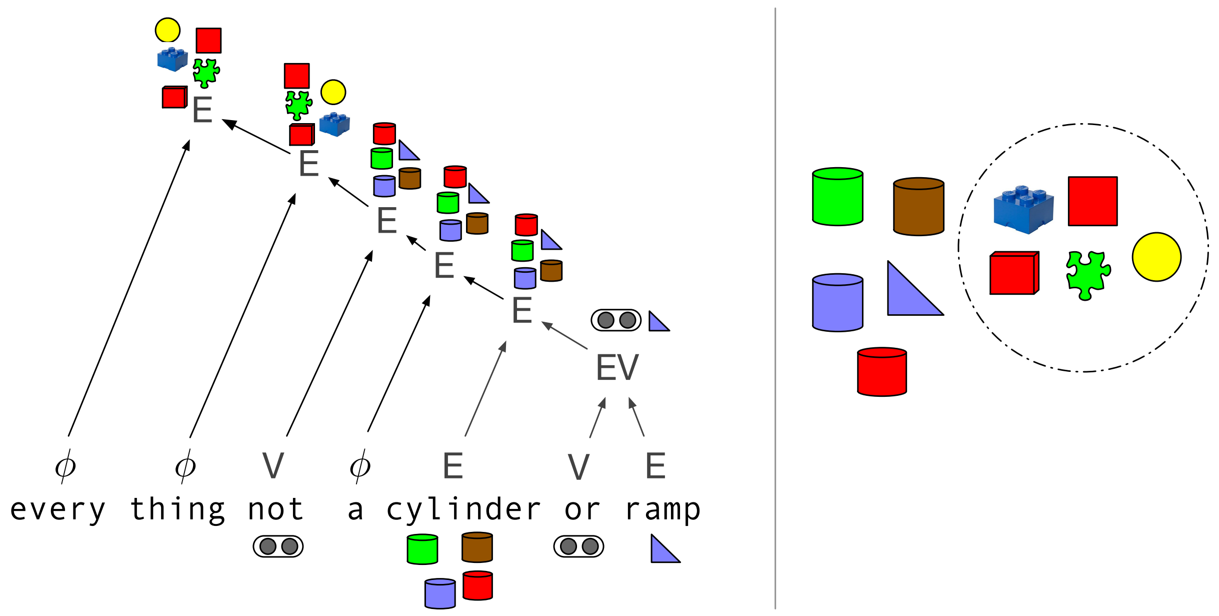
The semantic parsing model from (?) answers natural language queries for semi-structured tables. The answer is a denotation as a list of cells in the table. To use the SEMPRE framework, we convert the KGs in the GENX to tables as follows:

1. Each table has the first row (header) as: $\mid ObjId \mid P1 \mid P2 \mid P3 \mid P4 \mid$
2. Each row contains an object id, and the 4 property-attribute values in cells.
3. The answer denotation, i.e. the objects selected by the human annotators is now represented as a list of object ids.

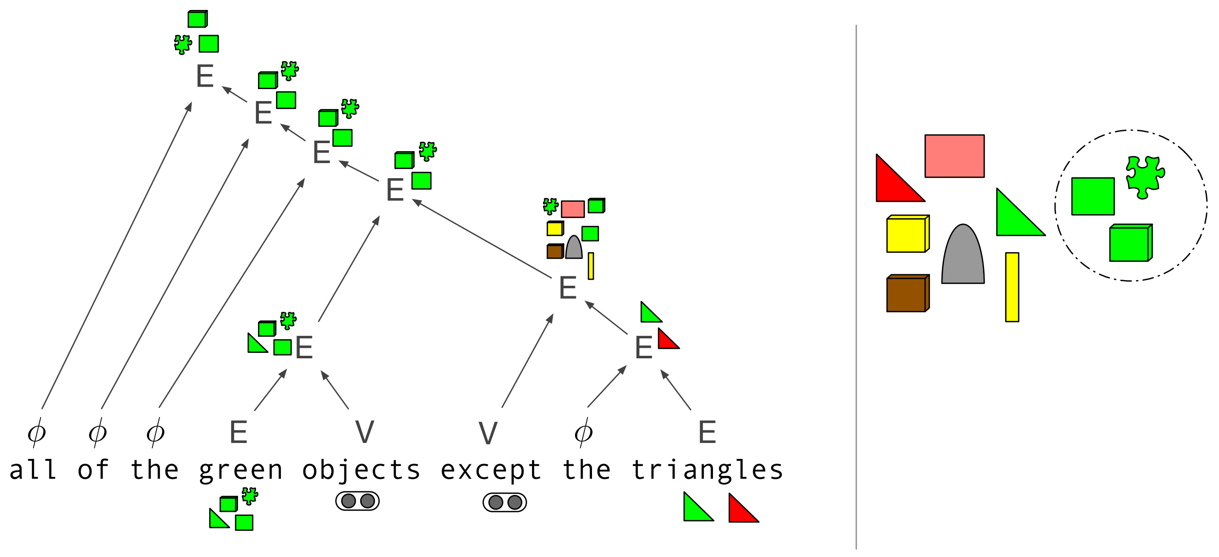
After converting the the KGs to tables, SEMPRE framework can be trivially used to train and test on the GENX dataset. We tune the number of epochs to train for based on the validation accuracy and find 8 epochs over the training data to work the best. We use the default setting of the other hyper-parameters.

D Example Output Parses

In Figure 1, we show example queries and their highest scoring output structure from our learned model for GENX dataset.



(a) An example output from our learned model showing that our model learns to correctly parse the questions sentence, and model the relevant semantic operator; *or* as a set union operation, to generate the correct answer denotation. It also learns to cope with lexical variability in human language; *triangle* being referred to as *ramp*.



(b) An example output from our learned model that shows that our model can learn to correctly parse human-generated language into relatively complex structures and model semantic operators, such as *except*, that were not encountered during model development.

Figure 1: **Example output structures from our learned model**: Examples of queries from the GenX dataset and the corresponding highest scoring tree structures from our learned model. The examples shows that our model is able to correctly parse human-generated language and jointly learn to model semantic operators, such as set unions, negations etc.