Weakly-supervised Neural Semantic Parsing with a Generative Ranker

Jianpeng Cheng and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
jianpeng.cheng@ed.ac.uk mlap@inf.ed.ac.uk

Abstract

Weakly-supervised semantic parsers are trained on utterance-denotation pairs, treating logical forms as latent. The task is challenging due to the large search space and spuriousness of logical forms. In this paper we introduce a neural parser-ranker system for weakly-supervised semantic parsing. The parser generates candidate tree-structured logical forms from utterances using clues of denotations. These candidates are then ranked based on two criterion: their likelihood of executing to the correct denotation, and their agreement with the utterance semantics. We present a scheduled training procedure to balance the contribution of the two objectives. Furthermore, we propose to use a neurally encoded lexicon to inject prior domain knowledge to the model. Experiments on three Freebase datasets demonstrate the effectiveness of our semantic parser, achieving results within the state-of-the-art range.

1 Introduction

Semantic parsing is the task of converting natural language utterances into machine-understandable meaning representations or logical forms. The task has attracted much attention in the literature due to a wide range of applications ranging from question answering (Kwiatkowski et al., 2011; Liang et al., 2011) to relation extraction (Krishnamurthy and Mitchell, 2012), goal-oriented dialog (Wen et al., 2015), and instruction understanding (Chen and Mooney, 2011; Matuszek et al., 2012; Artzi and Zettlemoyer, 2013).

In a typical semantic parsing scenario, a logical form is executed against a knowledge base to produce an outcome (e.g., an answer) known as denotation. Conventional semantic parsers are trained on collections of utterances paired with annotated logical forms (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2006; Kwiatkowksi et al., 2010). However, the labeling of logical forms is labor-intensive and challenging to elicit at a large scale. As a result, alternative forms of supervision have been proposed to alleviate the annotation bottleneck faced by semantic parsing systems. One direction is to train a semantic parser in a weakly-supervised setting based on utterance-denotation pairs (Clarke et al., 2010; Kwiatkowski et al., 2013; Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013), since such data are relatively easy to obtain via crowdsourcing (Berant et al., 2013a).

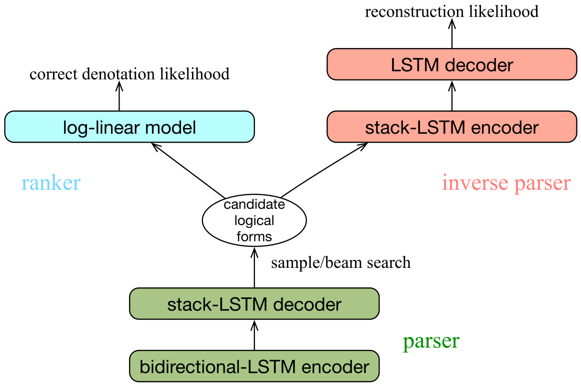


Figure 1: Overview of the weakly-supervised neural semantic parsing system.

However, the unavailability of logical forms in the weakly-supervised setting, renders model training more difficult. A fundamental challenge in learning semantic parsers from denotations is finding *consistent* logical forms, i.e., those which execute to the correct denotation. This search space can be very large, growing exponentially as compositionality increases. Moreover, consistent logical forms unavoidably introduce a certain degree of *spuriousness* — some of them will accidentally execute to the correct denotation without reflecting the meaning of the utterance. These spurious logical forms are misleading supervision sig-

Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018), pages 356–367
Brussels, Belgium, October 31 - November 1, 2018. ©2018 Association for Computational Linguistics

nals for the semantic parser.

In this work we introduce a weakly-supervised neural semantic parsing system which aims to handle both challenges. Our system, shown in Figure 1, mainly consists of a sequence-to-tree parser which generates candidate logical forms for a given utterance. These logical forms are subsequently ranked by two components: a log-linear model scores the likelihood of each logical form executing to the correct denotation, and an inverse neural parser measures the degree to which the logical form represents the meaning of the utterance. We present a scheduled training scheme which balances the contribution of the two components and objectives. To further boost performance, we propose to neurally encode a lexicon, as a means of injecting prior domain knowledge to the neural parameters.

We evaluate our system on three Freebase datasets which consist of utterance denotation pairs: WEBQUESTIONS (Berant et al., 2013a), GRAPHQUESTIONS (Su et al., 2016), and SPADES (Bisk et al., 2016). Experimental results across datasets show that our weakly-supervised semantic parser achieves state-of-the-art performance.

2 The Neural Parser-Ranker

Conventional weakly-supervised semantic parsers (Liang, 2016) consist of two major components: a parser, which is chart-based and non-parameterized, recursively builds derivations for each utterance span using dynamic programming. A learner, which is a log-linear model, defines features useful for scoring and ranking the set of candidate derivations, based on the correctness of execution results. As mentioned in Liang (2016), the chart-based parser brings a disadvantage since it does not support incremental contextual interpretation. The dynamic programming algorithm requires that features of a span are defined over sub-derivations in that span.

In contrast to a chart-based parser, a parameterized neural semantic parser decodes logical forms with global utterance features. However, training a weakly-supervised neural parser is challenging since there is no access to gold-standard logical forms for backpropagation. Besides, it should be noted that a neural decoder is conditionally generative: decoding is performed greedily conditioned on the utterance and the generation history—it makes no use of global logical form features. In this section, we introduce a parser-ranker framework which combines the best of conventional and neural approaches in the context of weakly-supervised semantic parsing.

2.1 Parser

Our work follows Cheng et al. (2017b, 2018) in using LISP-style functional queries as the logical formulation. Advantageously, functional queries are recursive, tree-structured and can naturally encode logical form derivations (i.e., functions and their application order). For example, the utterance "*who is obama's eldest daughter*" is simply represented with the function-argument structure `argmax(daughterOf(Obama), ageOf)`. Table 1 displays the functions we use in this work; a more detailed specifications can be found in the appendix.

To generate logical forms, our system adopts a variant of the neural sequence-to-tree model proposed in Cheng et al. (2017b). During generation, the prediction space is restricted by the grammar of the logical language (e.g., the type and the number of arguments required by a function) in order to ensure that output logical forms are well-formed and executable. The parser consists of a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) encoder and a stack-LSTM (Dyer et al., 2015) decoder, introduced as follows.

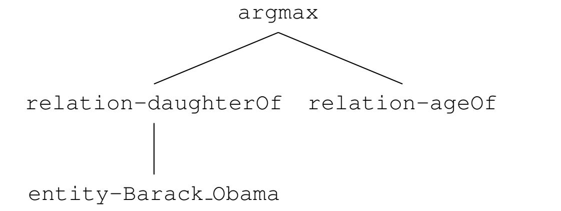
Bidirectional-LSTM Encoder The bidirectional LSTM encodes a variable-length utterance $x = (x_1, \cdots, x_n)$ into a list of token representations $[h_1, \cdots, h_n]$, where each representation is the concatenation of the corresponding forward and backward LSTM states.

Stack-LSTM Decoder After the utterance is encoded, the logical form is generated with a stack-LSTM decoder. The output of the decoder consists of functions which generate the logical form as a derivation tree in depth-first order. There are three classes of functions:

- *Class-1* functions generate non-terminal tree nodes. In our formulation, non-terminal nodes include language-dependent functions such as `count` and `argmax`, as described in the first four rows of Table 1. A special non-terminal node is the relation placeholder `relation`.
- *Class-2* functions generate terminal tree nodes. In our formulation, terminal nodes in-

Function	Utility	Example
`findAll`	returns the entity set of a given type	*find all mountains* `findAll(mountain)`
$\texttt{filter}_=$ $\texttt{filter}_<$ $\texttt{filter}_>$	filters an entity set with constraints	*all mountains in Europe* $\texttt{filter}_=$`(findAll(mountain), mountain_location, Europe)`
`count`	computes the cardinality of an entity set	*how many mountains are there* `count(findAll(mountain))`
`argmax` `argmin`	finds the subset of an entity set whose certain property is maximum (or minimum)	*the highest mountain* `argmax(findAll(mountain), mountain_altitude)`
`relation`	denotes a KB relation; in generation, `relation` acts as placeholder for all relations	*height of mountain* `mountain_altitude`
`entity`	denotes a KB entity; in generation, `entity` acts as placeholder for all entities	*Himalaya* `Himalaya`

Table 1: List of functions supported by our functional query language, their utility, and examples.



Functions for generation (parser): `argmax, relation, entity, reduce, relation, reduce`

Functions for encoding (inverse parser): `entity, relation, reduce, relation, argmax, reduce`

Figure 2: Derivation tree for the utterance "*who is obama's eldest daughter*" (top), and corresponding functions for generation and encoding (bottom).

clude the relation placeholder `relation` and the entity placeholder `entity`.

- *Class-3* function `reduce` completes a subtree. Since generation is performed in depth-first order, the parser needs to identify when the generation of a subtree completes, i.e., when a function has seen all its required arguments.

The functions used to generate the example logical form `argmax(daughterOf(Obama), ageOf)` are shown in Figure 2. The stack-LSTM makes two types of updates based on the functions it predicts:

- *Update-1*: when a *Class-1* or *Class-2* function is called, a non-terminal or terminal token l_t will be generated, At this point, the stack-LSTM state, denoted by g_t, is updated from its older state g_{t-1} as in an ordinary LSTM:

$$g_t = \text{LSTM}(l_t, g_{t-1}) \tag{1}$$

The new state is additionally pushed onto the stack marking whether it corresponds to a non-terminal or terminal.

- *Update-2*: when the `reduce` function is called (*Class-3*), the states of the stack-LSTM are recursively popped from the stack until a non-terminal is encountered. This non-terminal state is popped as well, after which the stack-LSTM reaches an intermediate state denoted by $g_{t-1:t}$. At this point, we compute the representation of the completed subtree z_t as:

$$z_t = W_z \cdot [p_z : c_z] \tag{2}$$

where p_z denotes the parent (non-terminal) embedding of the subtree, and c_z denotes the average embedding of the children (terminals or already-completed subtrees). W_z is the weight matrix. Finally, z_t serves as input for updating $g_{t-1:t}$ to g_t:

$$g_t = \text{LSTM}(z_t, g_{t-1:t}) \tag{3}$$

Prediction At each time step of the decoding, the parser first predicts a subsequent function f_{t+1} conditioned on the decoder state g_t and the encoder states $h_1 \cdots h_n$. We apply standard soft attention (Bahdanau et al., 2015) between g_t and the encoder states $h_1 \cdots h_n$ to compute a feature representation $\bar{h}_t$:

$$u_t^i = V \tanh(W_h h_i + W_g g_t) \tag{4}$$

$$a_t^i = \text{softmax}(u_t^i) \tag{5}$$

$$\bar{h}_t = \sum_{i=1}^{n} a_t^i h_i \tag{6}$$

where V, W_h, and W_g are all weight parameters. The prediction of the function f_{t+1} is computed with a softmax classifier, which takes the concatenated features $\bar{h}_t$ and g_t as input:

$$f_{t+1} \sim \mathrm{softmax}(W_y \tanh(W_f[\bar{h}_t, g_t])) \tag{7}$$

where W_y and W_f are weight parameters. When f_{t+1} is a language-dependent function (first four rows in Table 1, e.g., `argmax`), it is directly used as a non-terminal token l_{t+1} to construct the logical form. However, when f_{t+1} is a `relation` or `entity` placeholder, we further predict the specific relation or entity l_{t+1} with another set of neural parameters:

$$l_{t+1} \sim \mathrm{softmax}(W_{y'} \tanh(W_l[\bar{h}_t, g_t])) \tag{8}$$

where $W_{y'}$ and $W_{l'}$ are weight matrices.

Note that in the weakly supervised setting, the parser decodes a list of candidate logical forms Y with beam search, instead of outputting the most likely logical form y. During training, candidate logical forms are executed against a knowledge base to find those which are consistent (denoted by $Y_c(x)$) and lead to the correct denotation. Then, the parser is trained to maximize the total log likelihood of these consistent logical forms:

$$\begin{aligned} &\sum_{y \in Y_c(x)} \log p(y|x) = \\ &\sum_{y \in Y_c(x)} \log p(f_1, \cdots, f_k, l_1, \cdots, l_o|x) \end{aligned} \tag{9}$$

where k denotes the number of functions used to generate the logical form, and o (smaller than k) denotes the number of tree nodes in the logical form.

2.2 Ranker

It is impractical to rely solely on a neural decoder to find the most likely logical form at run time in the weakly-supervised setting. One reason is that although the decoder utilizes global utterance features for generation, it cannot leverage global features of the logical form since a logical form is conditionally generated following a specific tree-traversal order. To this end, we follow previous work (Berant et al., 2013b) and introduce a ranker to the system. The role of the ranker is to score the candidate logical forms generated by the parser; at test time, the logical form receiving the highest score will be used for execution. The ranker is a discriminative log-linear model over logical form y given utterance x:

$$\log_\theta p(y|x) = \frac{\exp(\phi(x,y)^T\theta)}{\sum_{y' \in Y(x)} \exp(\phi(x,y')^T\theta)} \tag{10}$$

where $Y(x)$ is the set of candidate logical forms; ϕ is the feature function that maps an utterance-logical form pair onto a feature vector; and θ denotes the weight parameters of the model.

Since the training data consists only of utterance-denotation pairs, the ranker is trained to maximize the log-likelihood of the correct answer z by treating logical forms as a latent variable:

$$\log p(z|x) = \log \sum_{y \in Y_c(x)} p(y|x)p(z|x,y) \tag{11}$$

where $Y_c(x)$ denotes the subset of candidate logical forms which execute to the correct answer; and $p(z|x,y)$ equates to 1 in this case.

Training of the neural parser-ranker system involves the following steps. Given an input utterance, the parser first generates a list of candidate logical forms via beam search. The logical forms are then executed and those which yield the correct denotation are marked as consistent. The parser is trained to optimize the total likelihood of consistent logical forms (Equation (9)), while the ranker is trained to optimize the marginal likelihood of denotations (Equation (11)). The search space can be further reduced by performing entity linking which restricts the number of logical forms to those containing only a small set of entities.

3 Handling Spurious Logical Forms

The neural parser-ranker system relies on beam search to find consistent logical forms that execute to the correct answer. These logical forms are then used as surrogate annotations and provide supervision to update the parser's parameters. However, some of these logical forms will be misleading training signals for the neural semantic parser on account of being spurious: they coincidentally execute to the correct answer without matching the utterance semantics.

In this section we propose a method of removing spurious logical forms by validating how well they match the utterance meaning. The intuition is that a meaning-preserving logical form should be able to reconstruct the original utterance with high likelihood. However, since spurious logical forms are not annotated either, a direct maximum likelihood solution does not exist. To this end, we propose a generative model for measuring the *reconstruction* likelihood.

The model assumes utterance x is generated from the corresponding logical form y, and only the utterance is observable. The objective is therefore to maximize the log marginal likelihood of x:

$$\log p(x) = \log \sum_{y} p(x, y) \tag{12}$$

We adopt neural variational inference (Mnih and Gregor, 2014) to solve the above objective, which is equivalent to maximizing an evidence lower bound:

$$\log p(x) = \log \frac{q(y|x)p(x|y)p(y)}{q(y|x)} \tag{13}$$
$$\geq \mathbb{E}_{q(y|x)} \log p(x|y) + \mathbb{E}_{q(y|x)} \log \frac{p(y)}{q(y|x)}$$

Since our semantic parser always outputs well-formed logical forms, we assume a uniform constant prior $p(y)$. The above objective can be thus reduced to:

$$\mathbb{E}_{q(y|x)} \log p(x|y) - \mathbb{E}_{q(y|x)} \log q(y|x) = \mathcal{L}(x) \tag{14}$$

where the first term computes the reconstruction likelihood $p(x|y)$; and the second term is the entropy of the approximated posterior $q(y|x)$ for regularization. Specifically, we use the semantic parser to compute the approximated posterior $q(y|x)$.[1] The reconstruction likelihood $p(x|y)$ is computed with an inverse parser which recovers utterance x from its logical form y. We use $p(x|y)$ to measure how well the logical form reflects the utterance meaning; details of the inverse parser are described as follows.

Stack-LSTM Encoder To reconstruct utterance x, logical form y is first encoded with a stack-LSTM encoder. To do that, we deterministically convert the logical form into a sequence of *Class-1* to *Class-3* functions, which correspond to the creation of tree nodes and subtrees. Slightly different from the top-down generation process, the functions here are obtained in a bottom-up order to facilitate encoding. Functions used to encode the example logical form `argmax(daughterOf(Obama), ageOf)` are shown in Figure 2.

The stack-LSTM sequentially processes the functions and updates its states based on the class of each function, following the same principle (*Update-1* and *Update-2*) described in Section 2.1. We save a list of terminal, non-terminal and subtree representations $[g_1, \cdots, g_s]$, where each representation is the stack-LSTM state at the corresponding time step of encoding. The list essentially contains the representation of every tree node and the representation of every subtree (the total number of representations is denoted by s).

LSTM Decoder Utterance x is reconstructed with a standard LSTM decoder attending to tree nodes and subtree representations. At each time step, the decoder applies attention between decoder state r_t and tree fragment representations $[g_1, \cdots, g_s]$:

$$v_t^i = V' \tanh(W_{g'} g_i + W_r r_t) \tag{15}$$

$$b_t^i = \text{softmax}(v_t^i) \tag{16}$$

$$\bar{g}_t = \sum_{i=1}^{s} b_t^i g_i \tag{17}$$

and predicts the probability of the next word as:

$$x'_{t+1} \sim \text{softmax}(W_{x'} \tanh(W_{f'}[\bar{g}_t, r_t])) \tag{18}$$

where Ws and V' are all weight parameters.

Gradients The training objective of the generative model is given in Equation (14). The parameters of the neural network include those of the original semantic parser (denoted by θ) and the inverse parser (denoted by ϕ). The gradient of Equation (14) with respect to ϕ is:

$$\frac{\partial \mathcal{L}(x)}{\partial \phi} = \mathbb{E}_{q(y|x)} \frac{\partial \log p(x|y)}{\partial \phi} \tag{19}$$

and the gradient with respect to θ is:

$$\frac{\partial \mathcal{L}(x)}{\partial \theta} = \mathbb{E}_{q(y|x)}[(\log p(x|y) - \log q(y|x)) \times \frac{\partial \log q(y|x)}{\partial \theta}] \tag{20}$$

Both gradients involve expectations which we estimate with Monte Carlo method, by sampling logical forms from the distribution $q(y|x)$. Recall that in the parser-ranker framework these samples are obtained via beam search.

[1] In Section 2.1, we used a different notation for the output distribution of the semantic parser as $p(y|x)$.

4 Scheduled Training

Together with the inverse parser for removing spurious logical forms, the proposed system consists of three components: a parser which generates logical forms from an utterance, a ranker which measures the likelihood of a logical form executing to the *correct denotation*, and an inverse parser which measures the degree to which logical forms are *meaning-preserving* using reconstruction likelihood. Our semantic parser is trained following a scheduled training procedure, balancing the two objectives.

- *Phase 1*: at the beginning of training when all model parameters are far from optimal, we train only the parser and the ranker as described in Section 2; the parser generates a list of candidate logical forms, we find those which are consistent and update both the parser and the ranker.

- *Phase 2*: we turn on the inverse parser and update all three components in one epoch. However, the reconstruction loss is only used to update the inverse parser and we prevent it from back-propagating to the semantic parser. This is because at this stage of training the parameters of the inverse parser are sub-optimal and we cannot obtain an accurate approximation of the reconstruction loss.

- *Phase 3*: finally, we allow the reconstruction loss to back-propagate to the parser, and all three components are updated as normal. Both training objectives are enabled, the system maximizes the likelihood of consistent logical forms and the reconstruction likelihood.

5 Neural Lexicon Encoding

In this section we further discuss how the semantic parser presented so far can be enhanced with a lexicon. A lexicon is essentially a coarse mapping between natural language phrases and knowledge base relations and entities, and has been widely used in conventional chart-based parsers (Berant et al., 2013a; Reddy et al., 2014). Here, we show how a lexicon (either hard-coded or statistically-learned (Krishnamurthy, 2016)) can be used to benefit a neural semantic parser.

The central idea is that relations or entities can be viewed as a single-node tree-structured logical form. For example, based on the lexicon, the natural language phrase "*is influenced by*" can be parsed to the logical form `influence.influence_node.influenced_by`. We can therefore pretrain the semantic parser (and the inverse parser) with these basic utterance-logical form pairs which act as important prior knowledge for initializing the distributions $q(y|x)$ and $p(x|y)$. With pre-trained word embeddings capturing linguistic regularities on the natural language side, we also expect the approach to help the neural model generalize to unseen natural language phrases quickly. For example, by encoding the mapping between the natural language phrase "*locate in*" and the Freebase predicate `fb:location.location.containedby`, the parser can potentially link the new phrase "*located at*" to the same predicate. We experimentally assess whether the neural lexicon enhances the performance of our semantic parser.

6 Experiments

In this section we evaluate the performance our semantic parser. We introduce the various datasets used in our experiments, training settings, model variants used for comparison, and finally present and analyze our results.

6.1 Datasets

We evaluated our model on three Freebase datasets: WEBQUESTIONS (Berant et al., 2013a), GRAPHQUESTIONS (Su et al., 2016) and SPADES (Bisk et al., 2016). WEBQUESTIONS contains 5,810 real questions asked by people on the web paired by answers. GRAPHQUESTIONS contains 5,166 question-answer pairs which were created by showing 500 Freebase graph queries to Amazon Mechanical Turk workers and asking them to paraphrase them into natural language. SPADES contains 93,319 question-answer pairs which were created by randomly replacing entities in declarative sentences with a blank symbol.

6.2 Training

Across training regimes, the dimensions of word vector, logical form token vector, and LSTM hidden states (for the semantic parser and the inverse parser) are 50, 50, and 150, respectively. Word embeddings were initialized with Glove embeddings (Pennington et al., 2014). All other embeddings were randomly initialized. We used one

LSTM layer in the forward and backward directions. Dropout was used before the softmax activation (Equations (7), (8), and (18)). The dropout rate was set to 0.5. Momentum SGD (Sutskever et al., 2013) was used as the optimization method to update the parameters of the model.

As mentioned earlier, we use entity linking to reduce the beam search space. Entity mentions in SPADES are automatically annotated with Freebase entities (Gabrilovich et al., 2013). For WEBQUESTIONS and GRAPHQUESTIONS we perform entity linking following the procedure described in Reddy et al. (2016). We identify potential entity spans using seven handcrafted part-of-speech patterns and associate them with Freebase entities obtained from the Freebase/KG API.[2] We use a structured perceptron trained on the entities found in WEBQUESTIONS and GRAPHQUESTIONS to select the top 10 non-overlapping entity disambiguation possibilities. We treat each possibility as a candidate entity and construct candidate utterances with a beam search of size 300.

Key features of the log-linear ranker introduced in Section 2 include the entity score returned by the entity linking system, the likelihood score of the relation in the logical form predicted by the parser, the likelihood score of the the logical form predicted by the parser, the embedding similarity between the relation in the logical form and the utterance, the similarity between the relation and the question words in the utterance, and the answer type as indicated by the last word in the Freebase relation (Xu et al., 2016). All features are normalized across candidate logical forms. For all datasets we use average F1 (Berant et al., 2013a) as our evaluation metric.

6.3 Model Variants

We experiment with three variants of our model. We primarily consider the neural parser-ranker system (denoted by NPR) described in Section 2 which is trained to maximize the likelihood of consistent logical forms. We then compare it to a system augmented with a generative ranker (denoted by GRANKER), introducing the second objective of maximizing the reconstruction likelihood. Finally, we examine the impact of neural lexicon encoding when it is used for the generative ranker, and also when it is used for the entire system.

[2] http://developers.google.com/freebase/

Models	F1
Berant et al. (2013a)	35.7
Berant and Liang (2014)	39.9
Berant and Liang (2015)	49.7
Reddy et al. (2016)	50.3
Yao and Van Durme (2014)	33.0
Bast and Haussmann (2015)	49.4
Bordes et al. (2014)	39.2
Dong et al. (2015)	40.8
Yih et al. (2015)	52.5
Xu et al. (2016)	53.3
Cheng et al. (2017b)	49.4
NPR	50.1
+ GRANKER	50.2
+ lexicon encoding on GRANKER	51.7
+ lexicon encoding on parser and GRANKER	52.5

Table 2: WEBQUESTIONS results.

6.4 Results

Experimental results on WEBQUESTIONS are shown in Table 2. We compare the performance of NPR with previous work, including conventional chart-based semantic parsing models (e.g., Berant et al. (2013a); first block in Table 2), information extraction models (e.g., Yao and Van Durme (2014); second block in Table 2), and more recent neural question-answering models (e.g., Dong et al. (2015); third block in Table 2). Most neural models do not generate logical forms but instead build a differentiable network to solve a specific task such as question-answering. An exception is the neural sequence-to-tree model of Cheng et al. (2017b), which we extend to build the vanilla NPR model. A key difference of NPR is that it employs soft attention instead of hard attention, which is Cheng et al. (2017b) use to rationalize predictions.

As shown in Table 2, the basic NPR system outperforms most previous chart-based semantic parsers. Our results suggest that neural networks are powerful tools for generating candidate logical forms in a weakly-supervised setting, due to their ability of encoding and utilizing sentential context and generation history. Compared to Cheng et al. (2017b), our system also performs better. We believe the reason is that it employs soft attention instead of hard attention. Soft attention makes the parser fully differentiable and optimization easier. The addition of the inverse parser (+GRANKER) to the basic NPR model yields marginal gains while

Models	F1
SEMPRE (Berant et al., 2013a)	10.80
PARASEMPRE (Berant and Liang, 2014)	12.79
JACANA (Yao and Van Durme, 2014)	5.08
SCANNER (Cheng et al., 2017b)	17.02
UDEPLAMBDA (Reddy et al., 2017)	17.70
NPR	17.30
+ GRANKER	17.33
+ lexicon encoding on GRANKER	17.67
+ lexicon encoding on parser and GRANKER	18.22

Table 3: GRAPHQUESTIONS results.

Models	F1
Unsupervised CCG (Bisk et al., 2016)	24.8
Semi-supervised CCG (Bisk et al., 2016)	28.4
Supervised CCG (Bisk et al., 2016)	30.9
Rule-based system (Bisk et al., 2016)	31.4
Sequence-to-tree (Cheng et al., 2017b)	31.5
Memory networks (Das et al., 2017)	39.9
NPR	32.4
+ GRANKER	33.1
+ lexicon encoding on GRANKER	35.5
+ lexicon encoding on parser and GRANKER	37.6

Table 4: SPADES results.

the addition of the neural lexicon encoding to the inverse parser brings performance improvements over NPR and GRANKER. We hypothesize that this is because the inverse parser adopts an unsupervised training objective, which benefits substantially from prior domain-specific knowledge used to initialize its parameters. When neural lexicon encoding is incorporated in the semantic parser as well, system performance can be further improved. In fact, our final system (last row in Table 2) outperforms all previous models except that of Xu et al. (2016), which uses external Wikipedia resources to prune out erroneous candidate answers.

Tables 3 and 4 present our results on GRAPHQUESTIONS and SPADES, respectively. Comparison systems for GRAPHQUESTIONS include two chart-based semantic parsers (Berant et al., 2013a; Berant and Liang, 2014), an information extraction model (Yao and Van Durme, 2014), a neural sequence-to-tree model with hard attention (Cheng et al., 2017b) and a model based on universal dependency to logical form conversion (Reddy et al., 2017). On SPADES we compare with the method of Bisk et al. (2016) which parses an utterance into a syntactic representation which is subsequently grounded to Freebase; and also with Das et al. (2017) who employ memory networks and external text resources. Results on both datasets follow similar trends as in WEBQUESTIONS. The best performing NPR variant achieves state-of-the-art results on GRAPHQUESTIONS and it comes close to the best model on SPADES without using any external resources.

One of the claims put forward in this paper is that the extended NPR model reduces the impact of spurious logical forms during training. Table 5 highlights examples of spurious logical forms which are not semantically correct but are nevertheless assigned higher scores in the vanilla NPR (red colour). These logical forms become less likely in the extended NPR, while the scores of more semantically faithful representations (blue colour) are boosted.

6.5 Discussion

The vanilla NPR model is optimized with consistent logical forms which lead to correct denotations. Although it achieves competitive results compared to chart-based parsers, the training of this model can be misled by spurious logical forms. The introduction of the inverse parser aims to alleviate the problem by scoring how a logical form reflects the utterance semantics. Although the inverse parser is not directly used to rank logical forms at test time, the training objective it adopts encourages the parser to generate meaning-preserving logical forms with higher likelihood. These probabilities are used as features in the log-linear ranker, and therefore the inverse parser affects the ranking results, albeit implicitly.

However, we should point out that the unsupervised training objective is relatively difficult to optimize, since there are no constraints to regularize the latent logical forms. This motivates us to develop a scheduled training procedure; as our results show, when trained properly the inverse parser and the unsupervised objective bring performance gains. Moreover, the neural lexicon encoding method we applied essentially produces synthetic data to further regularize the latent space.

7 Related Work

Various types of supervision have been explored to train semantic parsers. Early semantic parsers

which baseball teams were coached by dave eiland
baseball.batting_statistics.player:baseball.batting_statistics.team(ent.m.0c0x6v)
baseball.historical_coaching_tenure.baseball_coach:baseball.historical_coaching_tenure.baseball_team(ent.m.0c0x6v)

who are coca-cola's endorsers
food.nutrition_fact.food:food.nutrition_fact.nutrient(ent.m.01yvs)
business.product_endorsement.product:business..product_endorsement.endorser(ent.m.01yvs)

what are the aircraft models that are comparable to airbus 380
aviation.aviation_incident_aircraft_relationship.flight_destination:aviation.aviation_incident_aircraft_relationship.aircraft_model(ent.m.0qn2v)
aviation.comparable_aircraft_relationship(ent.m.018rl2)

Table 5: Comparison between logical forms preferred by NPR before and after the addition of the inverse parser. Spurious logical forms (red color) receive higher scores than semantically-correct ones (blue color). The scores of these spurious logical forms decrease when they are explicitly handled.

have used annotated training data consisting of sentences and their corresponding logical forms (Kate and Mooney, 2006; Kate et al., 2005; Lu et al., 2008; Kwiatkowksi et al., 2010). In order to scale semantic parsing to open-domain problems, weakly-supervised semantic parsers are trained on utterance-denotation pairs (Liang et al., 2011; Krishnamurthy and Mitchell, 2012; Berant et al., 2013b; Choi et al., 2015; Krishnamurthy and Mitchell, 2015; Pasupat and Liang, 2016; Gardner and Krishnamurthy, 2017; Reddy et al., 2017). Most previous work employs a chart-based parser to produce logical forms from a grammar which combines domain-general aspects with lexicons.

Recently, neural semantic parsing has attracted a great deal of attention. Previous work has mostly adopted fully-supervised, sequence-to-sequence models to generate logical form strings from natural language utterances (Dong and Lapata, 2016; Jia and Liang, 2016; Kočiský et al., 2016). Other work explores the use of reinforcement learning to train neural semantic parsers from question-answer pairs (Liang et al., 2016) or from user feedback (Iyer et al., 2017). More closely related to our work, Goldman et al. (2018) adopt a neural semantic parser and a discriminative ranker to solve a visual reasoning challenge. They attempt to alleviate the search space and spuriousness challenges with abstractive examples. Yin et al. (2018) adopt a tree-based variational autoencoder for semi-supervised semantic parsing. Neural variational inference has also been used in other NLP tasks including relation discovery (Marcheggiani and Titov, 2016), sentence compression (Miao and Blunsom, 2016), and parsing (Cheng et al., 2017a).

8 Conclusions

In this work we proposed a weakly-supervised neural semantic parsing system trained on utterance-denotation pairs. The system employs a neural sequence-to-tree parser to generate logical forms for a natural language utterance. The logical forms are subsequently ranked with two components and objectives: a log-linear model which scores the likelihood of correct execution, and a generative neural inverse parser which measures whether logical forms are meaning preserving. We proposed a scheduled training procedure to balance the two objectives, and a neural lexicon encoding method to initialize model parameters with prior knowledge. Experiments on three semantic parsing datasets demonstrate the effectiveness of our system. In the future, we would like to train our parser with other forms of supervision such as feedback from users (He et al., 2016; Iyer et al., 2017) or textual evidence (Yin et al., 2018).

Acknowledgments This research is supported by a Google PhD Fellowship and an AdeptMind Scolar Fellowship to the first author. We also gratefully acknowledge the financial support of the European Research Council (award number 681760; Lapata).

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 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 ICLR 2015*, San Diego, California.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440. ACM.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013a. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013b. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Yonatan Bisk, Siva Reddy, John Blitzer, Julia Hockenmaier, and Mark Steedman. 2016. Evaluating induced CCG parsers on grounded semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2022–2027, Austin, Texas.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 423–433, Sofia, Bulgaria.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial IntelligenceAAAI*, volume 2, pages 859–865, San Francisco, California.

Jianpeng Cheng, Adam Lopez, and Mirella Lapata. 2017a. A generative parser with a discriminative recognition algorithm. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 118–124, Vancouver, Canada.

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

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2018. Learning an executable neural semantic parser. *Computational Linguistics*.

Eunsol Choi, Tom Kwiatkowski, and Luke Zettlemoyer. 2015. Scalable semantic parsing with partial ontologies. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1311–1320, Beijing, China.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the 14th Conference on Computational Natural Language Learning*, pages 18–27, Uppsala, Sweden.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 358–365.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over Freebase with multicolumn convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 260–269, Beijing, China.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

Matt Gardner and Jayant Krishnamurthy. 2017. Open-Vocabulary Semantic Parsing with both Distributional Statistics and Formal Knowledge. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3195–3201, San Francisco, California.

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. 2018. Weakly supervised semantic parsing with abstract examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1809–1819, Melbourne, Australia.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2337–2342, Austin, Texas.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada.

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

Rohit J Kate and Raymond J Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 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 for the 20th National Conference on Artificial Intelligence*, pages 1062–1068, Pittsburgh, Pennsylvania.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087, Austin, Texas.

Jayant Krishnamurthy. 2016. Probabilistic models for learning a semantic parser lexicon. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 606–616.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765, Jeju Island, Korea.

Jayant Krishnamurthy and Tom M Mitchell. 2015. Learning a compositional semantics for freebase with an open predicate vocabulary. *Transactions of the Association for Computational Linguistics*, 3:257–270.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233, Cambridge, MA.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA.

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

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Communications of the ACM*, 59(9):68–76.

Percy Liang, Michael 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*, pages 590–599, Portland, Oregon.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 783–792, Honolulu, Hawaii.

Diego Marcheggiani and Ivan Titov. 2016. Discrete-state variational autoencoders for joint discovery and factorization of relations. *Transactions of the Association for Computational Linguistics*, 4:231–244.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1671–1678, Edinburgh, Scotland.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328, Austin, Texas.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1791–1799, Bejing, China.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 23–32, Berlin, Germany.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

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.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101, Copenhagen, Denmark.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572, Austin, Texas.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1139–1147, Atlanta, Georgia.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on Freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2326–2336, Berlin, Germany.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966, Baltimore, Maryland.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. StructVAE: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 754–765, Melbourne, Australia.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the 13th National Conference on Artificial Intelligence*, 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 21st Conference in Uncertainilty in Artificial Intelligence*, pages 658–666, Edinburgh, Scotland.