Enforcing Consistency in Weakly Supervised Semantic Parsing

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

Sameer Singh
University of California, Irvine
sameer@uci.edu

Matt Gardner
Allen Institute for AI
mattg@allenai.org

Abstract

The predominant challenge in weakly supervised semantic parsing is that of *spurious programs* that evaluate to correct answers for the wrong reasons. Prior work uses elaborate search strategies to mitigate the prevalence of spurious programs; however, they typically consider only one input at a time. In this work we explore the use of consistency between the output programs for related inputs to reduce the impact of spurious programs. We bias the program search (and thus the model's training signal) towards programs that map the same phrase in related inputs to the same sub-parts in their respective programs. Additionally, we study the importance of designing logical formalisms that facilitate this kind of consistency-based training. We find that a more consistent formalism leads to improved model performance even without consistency-based training. When combined together, these two insights lead to a 10% absolute improvement over the best prior result on the Natural Language Visual Reasoning dataset.

1 Introduction

Semantic parsers map a natural language utterance into an executable meaning representation, called a logical form or program (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005). These programs can be executed against a context (e.g., database, image, etc.) to produce a denotation (e.g., answer) for the input utterance. Methods for training semantic parsers from only (utterance, denotation) supervision have been developed (Clarke et al., 2010; Liang et al., 2011; Berant et al., 2013); however, training from such weak supervision is challenging. The parser needs to search for the correct program from an exponentially large space, and the presence of *spurious programs*—incorrect representations that evaluate to the correct denotation—greatly hampers learning. Several strategies have been proposed to mitigate this issue (Guu et al., 2017; Liang et al., 2018; Dasigi et al., 2019). Typically these approaches consider a single input utterance at a time and explore ways to score programs.

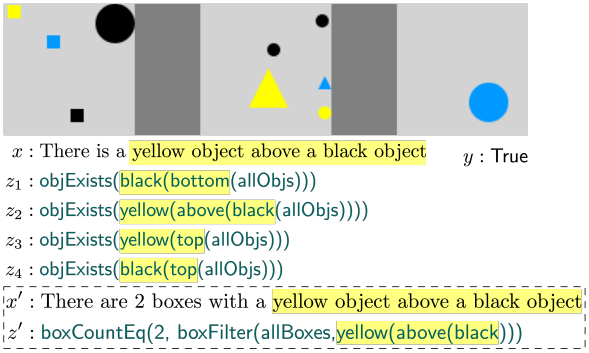

Figure 1: Utterance x and its program candidates z_1-z_4, all of which evaluate to the correct denotation (True). z_2 is the correct interpretation; other programs are *spurious*. Related utterance x' shares the phrase *yellow object above a black object* with x. Our consistency reward would score z_2 the highest since it maps the shared phrase most similarly compared to z'.

In this work we encourage consistency between the output programs of related natural language utterances to mitigate the issue of spurious programs. Consider related utterances, *There are two boxes with three yellow squares* and *There are three yellow squares*, both containing the phrase *three yellow squares*. Ideally, the correct programs for the utterances should contain similar sub-parts that corresponds to the shared phrase. To incorporate this intuition during search, we propose a consistency-based reward to encourage programs for related utterances that share sub-parts corresponding to the shared phrases (§3). By doing so, the model is provided with an additional training signal to distinguish between programs based on their consistency with programs predicted for related utterances.

[*] Work done while interning with Allen Institute for AI.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Short Papers), pages 168–174
August 1–6, 2021.

We also show the importance of designing the logical language in a manner such that the ground-truth programs for related utterances are consistent with each other. Such consistency in the logical language would facilitate the consistency-based training proposed above, and encourage the semantic parser to learn generalizable correspondence between natural language and program tokens. In the previously proposed language for the Natural Language Visual Reasoning dataset (NLVR; Suhr et al., 2017), we notice that the use of macros leads to inconsistent interpretations of a phrase depending on its context. We propose changes to this language such that a phrase in different contexts can be interpreted by the same program parts (§4).

We evaluate our proposed approaches on NLVR using the semantic parser of Dasigi et al. (2019) as our base parser. On just replacing the old logical language for our proposed language we see an 8% absolute improvement in consistency, the evaluation metric used for NLVR (§5). Combining with our consistency-based training leads to further improvements; overall 10% over the best prior model, reporting a new state-of-the-art on the NLVR dataset.

2 Background

In this section we provide a background on the NLVR dataset (Suhr et al., 2017) and the semantic parser of Dasigi et al. (2019).

Natural Language Visual Reasoning (NLVR) dataset contains human-written natural language utterances, where each utterance is paired with 4 synthetically-generated images. Each (utterance, image) pair is annotated with a binary truth-value denotation denoting whether the utterance is true for the image or not. Each image is divided into three *boxes*, where each box contains 1-8 *objects*. Each object has four properties: *position* (x/y coordinates), *color* (black, blue, yellow), *shape* (triangle, square, circle), and *size* (small, medium, large). The dataset also provides a structured representation of each image which we use in this paper. Figure 1 shows an example from the dataset.

Weakly supervised iterative search parser We use the semantic parser of Dasigi et al. (2019) which is a grammar-constrained encoder-decoder with attention model from Krishnamurthy et al. (2017). It learns to map a natural language utterance x into a program z such that it evaluates to the correct denotation $y = [\![z]\!]^r$ when executed against the structured image representation r. Dasigi et al. (2019) use a manually-designed, typed, variable-free, functional query language for NLVR, inspired by the GeoQuery language (Zelle and Mooney, 1996).

Given a dataset of triples (x_i, c_i, y_i), where x_i is an utterance, c_i is the set of images associated to it, and y_i is the set of corresponding denotations, their approach iteratively alternates between two phases to train the parser: Maximum marginal likelihood (MML) and a Reward-based method (RBM). In MML, for an utterance x_i, the model maximizes the marginal likelihood of programs in a given set of logical forms Z_i, all of which evaluate to the correct denotation. The set Z_i is constructed either by performing a heuristic search, or generated from a trained semantic parser.

The reward-based method maximizes the (approximate) expected value of a reward function $\mathcal{R}$.

$$\max_{\theta} \sum_{\forall i} \mathbb{E}_{\tilde{p}(z_i|x_i;\theta)} \mathcal{R}(x_i, z_i, c_i, y_i) \qquad (1)$$

Here, $\tilde{p}$ is the *re-normalization* of the probabilities assigned to the programs on the beam, and the reward function $\mathcal{R} = 1$ if z_i evaluates to the correct denotation for all images in c_i, or 0 otherwise. Please refer Dasigi et al. (2019) for details.

3 Consistency reward for programs

Consider the utterance x = *There is a yellow object above a black object* in Figure 1. There are many program candidates decoded in search that evaluate to the correct denotation. Most of them are *spurious*, i.e., they do not represent the meaning of the utterance and only coincidentally evaluate to the correct output. The semantic parser is expected to distinguish between the correct program and spurious ones by identifying correspondence between parts of the utterance and the program candidates. Consider a related utterance x' = *There are 2 boxes with a yellow object above a black object*. The parser should prefer programs for x and x' which contain similar sub-parts corresponding to the shared phrase p = *yellow object above a black object*. That is, the parser should be consistent in its interpretation of a phrase in different contexts. To incorporate this intuition during program search, we propose an additional reward to programs for an utterance that are consistent with programs for a related utterance.

Specifically, consider two related utterances x and x' that share a phrase p. We compute a reward for a program candidate z of x based on how similarly it maps the phrase p as compared to a program candidate z' of x'. To compute this reward we need (a) *relevant program parts* in z and z' that correspond to the phrase p, and (b) a *consistency reward* that measures consistency between those parts.

(a) Relevant program parts Let us first see how to identify relevant parts of a program z that correspond to a phrase p in the utterance.

Our semantic parser (from Krishnamurthy et al. (2017)) outputs a linearized version of the program $z = [z^1, \ldots, z^T]$, decoding one action at a time from the logical language. At each time step, the parser predicts a normalized attention vector over the tokens of the utterance, denoted by $[a_1^t, \ldots, a_N^t]$ for the z^t action. Here, $\sum_{i=1}^{N} a_i^t = 1$ and $a_i^t \geq 0$ for $i \in [1, N]$. We use these attention values as a relevance score between a program action and the utterance tokens. Given the phrase p with token span $[m, n]$, we identify the relevant actions in z as the ones whose total attention score over the tokens in p exceeds a heuristically-chosen threshold $\tau = 0.6$.

$$A(z, p) = \left\{ z^t \;\middle|\; t \in [1, T] \text{ and } \sum_{i=m}^{n} a_i^t \geq \tau \right\} \quad (2)$$

This set of program actions $A(z, p)$ is considered to be generated due to the phrase p. For example, for utterance *There is a yellow object above a black object*, with program objExists(yellow(above(black(allObjs)))), this approach could identify that for the phrase *yellow object above a black object* the actions corresponding to the functions yellow, above, and black are relevant.

(b) Consistency reward Now, we will define a reward for the program z based on how consistent its mapping of the phrase p is w.r.t. the program z' of a related utterance. Given a related program z' and its relevant action set $A(z', p)$, we define the consistency reward $S(z, z', p)$ as the F1 score for the action set $A(z, p)$ when compared to $A(z', p)$. If there are multiple shared phrases p_i between x and x', we can compute a weighted average of different $S(z, z', p_i)$ to compute a singular consistency reward $S(z, z')$ between the programs z and z'. In this work, we only consider a single shared phrase p between the related utterances, hence $S(z, z', p) = S(z, z', p)$ in our paper.

As we do not know the gold program for x', we decode top-K program candidates using beam-search and discard the ones that do not evaluate to the correct denotation. We denote this set of programs by Z'_c. Now, to compute a consistency reward $\mathcal{C}(x, z, x')$ for the program z of x,we take a weighted average of $S(z, z')$ for different $z' \in Z'_c$ where the weights correspond to the probability of the program z' as predicted by the parser.

$$\mathcal{C}(x, z, x') = \sum_{z' \in Z'_c} \tilde{p}(z'|x'; \theta) S(z, z') \quad (3)$$

Consistency reward based parser Given x and a related utterance x', we use $\mathcal{C}(x, z, x')$ as an additional reward in Eq. 1 to upweight programs for x that are consistent with programs for x'.

$$\max_{\theta} \sum_{\forall i} \mathbb{E}_{\tilde{p}(z_i|x_i;\theta)} \left[\mathcal{R}(x_i, z_i, c_i, y_i) + \mathcal{C}(x_i, z_i, x'_i) \right]$$

This consistency-based reward pushes the parser's probability mass towards programs that have consistent interpretations across related utterances, thus providing an additional training signal over simple denotation accuracy. The formulation presented in this paper assumes that there is a single related utterance x' for the utterance x. If multiple related utterances are considered, the consistency reward $\mathcal{C}(x, z, x'_j)$ for different related utterances x'_j can be summed/averaged to compute a single consistency reward $\mathcal{C}(x, z)$ the program z of utterance x based on all the related utterances.

4 Consistency in Language

The consistency reward (§3) makes a key assumption about the logical language in which the utterances are parsed: that the gold programs for utterances sharing a natural language phrase actually correspond to each other. For example, that the phrase *yellow object above a black object* would always get mapped to yellow(above(black)) irrespective of the utterance it occurs in.

On analyzing the logical language of Dasigi et al. (2019), we find that this assumption does not hold true. Let us look at the following examples:

x_1: *There are items of at least two different colors*
z_1: objColorCountGrtEq(2, allObjs)
x_2: *There is a box with items of at least two different colors*
z_2: boxExists(
memberColorCountGrtEq(2, allBoxes))

Here the phrase *items of at least two different colors*

Model	Dev		Test-P		Test-H	
	Acc.	Cons.	Acc.	Cons.	Acc.	Cons.
ABS. SUP. (Goldman et al., 2018)	84.3	66.3	81.7	60.1	-	-
ABS. SUP. + RERANK (Goldman et al., 2018)	85.7	67.4	84.0	65.0	82.5	63.9
ITERATIVE SEARCH (Dasigi et al., 2019)	85.4	64.8	82.4	61.3	82.9	64.3
+ Logical Language Design (ours)	88.2	73.6	86.0	69.6	-	-
+ Consistency Reward (ours)	**89.6**	**75.9**	**86.3**	**71.0**	**89.5**	**74.0**

Table 1: **Performance on NLVR:** Design changes in the logical language and consistency-based training, both significantly improve performance. Larger improvements in consistency indicate that our approach efficiently tackles spurious programs.

is interpreted differently in the two utterances. In x_2, a macro function memberColorCountGrtEq is used, which internally calls objColorCountGrtEq for each *box* in the image. Now consider,
x_3: *There is a tower with exactly one block*
z_3: boxExists(memberObjCountEq(1,allBoxes))
x_4: *There is a tower with a black item on the top*
z_4: objExists(black(top(allObjs)))
Here the phrase *There is a tower* is interpreted differently: z_3 uses a macro for filtering boxes based on their object count and interprets the phrase using boxExists. In the absence of a complex macro for checking *black item on the top*, z_4 resorts to using objExists making the interpretation of the phrase inconsistent. These examples highlight that these macros, while they shorten the search for programs, make the language inconsistent.

We make the following changes in the logical language to make it more consistent. Recall from §2 that each NLVR image contains 3 boxes each of which contains 1-8 objects. We remove macro functions like memberColorCountGrtEq, and introduce a generic boxFilter function. This function takes two arguments, a set of *boxes* and a filtering function f: Set[Obj] → bool, and prunes the input set of boxes to the ones whose objects satisfies the filter f. By doing so, our language is able to reuse the same object filtering functions across different utterances. In this new language, the gold program for the utterance x_2 would be
z_2: boxCountEq(1, boxFilter(allBoxes, objColorCountGrtEq(2)))
By doing so, our logical language can now consistently interpret the phrase *items of at least two different colors* using the object filtering function f: objColorCountGrtEq(2) across both x_1 and x_2. Similarly, the gold program for x_4 in the new logical language would be
z_4: boxExists(boxFilter(allBoxes, black(top)))
making the interpretation of *There is a box* consistent with x_3. Please refer appendix §A for details.

5 Experiments

Dataset We report results on the standard development, public-test, and hidden-test splits of NLVR. The training data contains 12.4k (utterance, image) pairs where each of 3163 utterances are paired with 4 images. Each evaluation set roughly contains 270 unique utterances.

Evaluation Metrics (1) *Accuracy* measures the proportion of examples for which the correct denotation is predicted. (2) Since each utterance in NLVR is paired with 4 images, a *consistency* metric is used, which measures the proportion of utterances for which the correct denotation is predicted for all associated images. Improvement in this metric is indicative of correct program prediction as it is unlikely for a spurious program to correctly make predictions on multiple images.

Experimental details We use the same parser, training methodology, and hyper-parameters as Dasigi et al. (2019). For discovering related utterances, we manually identify ∼10 sets of equivalent phrases that are common in NLVR. For example, *there are* NUM *boxes*, COLOR1 *block on a* COLOR2 *block*, etc. For each utterance that contains a particular phrase, we pair it with one other randomly chosen utterance that shares the phrase. We make 1579 utterance pairs in total. Refer appendix §B for details about data creation.[1]

Baselines We compare against the state-of-the-art models; ABS. SUP. (Goldman et al., 2018) that

[1]We release the data and code at https://www.github.com/nitishgupta/allennlp-semparse/tree/nlvr-v2/scripts/nlvr_v2

uses abstract examples, ABS. SUP. + RERANK that uses additional data and reranking, and the iterative search parser of Dasigi et al. (2019).

Results Table 1 compares the performance of our two proposed methods to enforce consistency in the decoded programs with the previous approaches. We see that changing the logical language to a more consistent one (§4) significantly improves performance: the accuracy improves by 2-4% and consistency by 4-8% on the dev. and public-test sets. Additionally, training the parser using our proposed consistency reward (§3) further improves performance: accuracy improves by 0.3-0.4% but the consistency significantly improves by 1.4-2.3%.[2] On the hidden-test set of NLVR, our final model improves accuracy by 7% and consistency by 10% compared to previous approaches. Larger improvements in consistency across evaluation sets indicates that our approach to enforce consistency between programs of related utterances greatly reduces the impact of spurious programs.

6 Conclusion

We proposed two approaches to mitigate the issue of spurious programs in weakly supervised semantic parsing by enforcing consistency between output programs. First, a consistency based reward that biases the program search towards programs that map the same phrase in related utterances to similar sub-parts. Such a reward provides an additional training signal to the model by leveraging related utterances. Second, we demonstrate the importance of logical language design such that it facilitates such consistency-based training. The two approaches combined together lead to significant improvements in the resulting semantic parser.

Acknowledgement

We would like to thank Pradeep Dasigi for helping us with the code for preprocessing NLVR and the Iterative Search model, Alane Suhr for getting us our model's evaluation results on the hidden test set in a timely manner, and the anonymous reviewers for their helpful comments. This work is supported in part by NSF award #IIS-1817183.

[2]We report average performance across 10 runs trained with different random seeds. All improvements in consistency are statistically significant (p-value < 0.05) based on the stochastic ordering test (Dror et al., 2019).

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from Question-Answer pairs. In *EMNLP*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL*.

Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke Zettlemoyer, and E. Hovy. 2019. Iterative search for weakly supervised semantic parsing. In *NAACL-HLT*.

Rotem Dror, Segev Shlomov, and Roi Reichart. 2019. Deep dominance-how to properly compare deep neural models. In *ACL*.

Omer Goldman, Veronica Latcinnik, Udi Naveh, A. Globerson, and Jonathan Berant. 2018. Weakly-supervised semantic parsing with abstract examples. In *ACL*.

Kelvin Guu, Panupong Pasupat, E. Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *ACL*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V. Le, and N. Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *NeurIPS*.

Percy S. Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. *Computational Linguistics*.

Alane Suhr, M. Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *ACL*.

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

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI '05*.

A Logical language details

In Figure 2, we show an example utterance with its gold program according to our proposed logical language. We use function composition and function currying to maintain the variable-free nature of our language. For example, action z^7 uses function composition to create a function from Set[Object] → bool by composing two functions, from Set[Object] → bool and Set[Object] → Set[Object]. Similarly, action z^{11} creates a function from Set[Object] → Set[Object] by composing two functions with the same signature.

Actions z^8 - z^{10} use function currying to curry the 2-argument function objectCountGtEq by giving it one int=2 argument. This results in a 1-argument function objectCountGtEq(2) from Set[Object] → bool.

B Dataset details

To discover related utterance pairs within the NLVR dataset, we manually identify 11 sets of phrases that commonly occur in NLVR and can be interpreted in the same manner:

1. { COLOR block at the base, the base is COLOR }
2. { COLOR block at the top, the top is COLOR }
3. { COLOR1 object above a COLOR2 object }
4. { COLOR1 block on a COLOR2 block, COLOR1 block over a COLOR2 block }
5. { a COLOR tower }
6. { there is one tower, there is only one tower, there is one box, there is only one box }
7. { there are exactly NUMBER towers, there are exactly NUMBER boxes }
8. { NUMBER different colors }
9. { with NUMBER COLOR items, with NUMBER COLOR blocks, with NUMBER COLOR objects }
10. { at least NUMBER COLOR items, at least NUMBER COLOR blocks, at least NUMBER COLOR objects }
11. { with NUMBER COLOR SHAPE, are NUMBER COLOR SHAPE, with only NUMBER COLOR SHAPE, are only NUMBER COLOR SHAPE }

In each phrase, we replace the abstract COLOR, NUMBER, SHAPE token with all possible options from the NLVR dataset to create grounded phrases. For example, *black block at the top*, *yellow object above a blue object*. For each set of equivalent grounded phrases, we identify the set of utterances that contains any of the phrase. For each utterance in that set, we pair it with 1 randomly chosen utterance from that set. Overall, we identify related utterances for 1420 utterances (out of 3163) and make 1579 pairings in total; if an utterance contains two phrases of interest, it can be paired with more than 1 utterance.

x: *There is one box with at least 2 yellow squares*
z: boxCountEq(1, boxFilter(allBoxes, objectCountGtEq(2)(yellow(square))))

Program actions for z:
z^1: bool → [<int,[Set[Box]:bool>, int, Set[Box]]
z^2: <int,[Set[Box]:bool> → boxCountEq
z^3: int → 1
z^4: Set[Box] → [<Set[Box],<Set[Object]:bool>:Set[Box]>, Set[Box], <Set[Object]:bool>]
z^5: <Set[Box],<Set[Object]:bool>:Set[Box]> → boxFilter
z^6: Set[Box] → allBoxes
z^7: <Set[Object]:bool> → [*, <Set[Object]:bool>, <Set[Object]:Set[Object]>]
z^8: <Set[Object]:bool> → [<int,Set[Object]:bool>, int]
z^9: <int,Set[Object]:bool> → objectCountGtEq
z^{10}: int → 2
z^{11}: <Set[Object]:Set[Object]> → [*, <Set[Object]:Set[Object]>, <Set[Object]:Set[Object]>]
z^{12}: <Set[Object]:Set[Object]> → yellow
z^{13}: <Set[Object]:Set[Object]> → square

Figure 2: Gold program actions for the utterance *There is one box with at least 2 yellow squares* according to our proposed logical language. The grammar-constrained decoder outputs a linearized abstract-syntax tree of the program in an in-order traversal.