Tackling Spuriousness with Similarity

Aran Carmon , Dor Muhlgay , Alon Resler , Tomer Wolfson
Blavatnik School of Computer Science, Tel-Aviv University
{arancarmon,dormuhlg,alonresler,tomerwol}@mail.tau.ac.il

1 Introduction

The goal of Semantic Parsing is to translate natural language utterances into logical forms that are executed against a target knowledge base. Ideally one would train the parser on the utterances and their annotations in formal language. However, annotating logical forms is a costly operation so work in recent years has suggested training the parser on examples labeled only with their correct output, and not with the correct formula that produced it (Clarke et al., 2010; Liang et al., 2011). This weak supervision facilitates the training process but comes at the price of learning from *spurious formulas*, logical forms which execute to the correct output but do not capture the meaning of the original utterance. Consider Table 1 and the utterance *"When did China host the Olympics?"*, its correct logical form should be *"value of Year where Country is China"*, although the logical form *"first value of Year after 2004"* will also yield the correct output, 2008. Both logical forms are *consistent* (execute to the correct output) but the latter is clearly spurious.

Year	Country	City
2004	Greece	Athens
2008	China	Beijing
2012	UK	London

Table 1: Knowledge base for question x = *"When did China host the Olympics?"*. Target denotation is y = {2008}.

Guu et al. (2017) show that since the gradient computation depends on the current model, if the correct logical form has low probability under this model then gradient updates upweight the distribution of spurious formulas more than that of correct ones. To address this, they propose a parameter update rule which more equally upweights all consistent formulas. Our approach is to define an update rule that upweights consistent logical forms based on their similarity to the original utterance. It is highly reasonable that the logical form most similar to the natural language utterance will also be its correct translation. To capture similarity, we implement a model for *paraphrastic sentence embeddings* (Wieting et al., 2016). We integrate our trained similarity model into a semantic parser and use it as part of the gradient computation. Our semantic parser is based on the floating beam-search parser presented in Pasupat and Liang (2015), henceforth PL2015.

2 Task

We consider the task of semantic parsing on semi-structured tables using the Wikitable Questions dataset (PL2015). The dataset contains 2,000 Wikipedia tables and over 20,000 question-answer pairs. Given a question in natural language, the parser aims to translate it into a logical form in lambda-DCS (Liang, 2013) that executes to the correct answer. Training exclusively on question-answer pairs results in weak supervision and in learning from spurious formulas, which we aim to reduce. Our **evaluation metric** is *accuracy*, defined as the percentage of examples for which the top ranked logical form is consistent.

3 System Overview

In training our semantic parser, we go through the following stages: (i) First, we generate candidate logical forms using the parser of Zhang et al. (2017); (ii) for each consistent logical form we compute its similarity to the original question using a pre-trained paraphrase model; (iii) the similarity scores are used to compute the formulas' rewards; (iv) finally, the

rewards are used to compute the parser's gradient update.

4 Similarity-based Rewards

4.1 Measuring Similarity

We wish to reward logical forms that are more similar to the original utterance. To measure such similarity we implemented a sentence embedding model for paraphrasing tasks (Wieting et al., 2016), which maps each sentence to a real vector.

4.2 Canonical Utterances

Our sentence paraphrase models are built for capturing similarity between two sentences rather than an utterance and formula. Instead of providing the logical forms themselves, we provide the model with their *canonical utterances* (Wang et al., 2015). These are natural language paraphrases of the logical forms, generated by augmenting the rules of our parser's grammar. For example, the formula:

$$\boldsymbol{R}[Year].City.Athens^{1}$$

is coupled with, *"Year where the City is Athens"*.

4.3 Models

We implement two similarity models. The first is a **Bag of Words** model, which embeds a word sequence $x = x_1, x_2, \cdots, x_n$ by averaging the vectors of its tokens. The only parameters learned by this model are the word embedding matrix W_w:

$$g(x) = \frac{1}{n}\sum_{i}^{n} W_w^{x_i}$$

where $W_w^{x_i}$ is the word vector of token x_i. This model uses parameters that are pre-trained on the Paraphrase Database (PPDB). It is limited by its disregard for word order and context in addition to being pre-trained on the PPDB dataset rather than on our data. The second model is a **LSTM** model trained directly on our data. Full details of the LSTM model are provided in the appendix.

[1] where **R[]** is the reverse of a relation

4.4 Rewards

Given question x, set of consistent formulas $Z^+ = z_1, \cdots, z_m$ and similarity scores $S = s(z_1, x), \cdots, s(z_m, x)$, we use two methods to set the formulas' rewards. Our first reward function uses softmax on the logical form's similarity vector:

$$R_{soft}(z_i) \propto exp\{c \cdot s(z_i, x)\}$$

where constant c is used to increase rewards variance. The second reward function eliminates $\lfloor p \cdot m \rfloor$ of the least similar formulas:

$$R_{elim}(z_i) = \begin{cases} 0 & \text{if } s_i(z_i, x) \in \text{ lowest } \lfloor p \cdot m \rfloor \text{ in } S \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

where p is the elimination ratio. For *inconsistent formulas* (i.e wrong output) rewards are set to 0.

5 Semantic Parser Training

5.1 Baseline Model and Learning

Our work is based on the *floating parser* presented in Pasupat and Liang (2015); Zhang et al. (2017). The parser learns a parameter vector θ from training on triplets $(x_i, w_i, y_i)_{i=1}^n$ of utterance x, knowledge graph w, and target denotation y. It constructs a set of candidate logical forms Z_i comprised of the sets of consistent and inconsistent logical forms, denoted Z_i^+ and Z_i^- respectively. For each $z \in Z_i$ the parser extracts a feature vector $\phi(x, w, z)$ and defines a log-linear distribution over all formulas in Z_i:

$$p_\theta(z \mid Z_i) \propto \exp\{\phi(z)^\top \theta\} \tag{2}$$

Enumerating over all logical forms in Z_i might prove infeasible so it is common practice to use beam search on the space of logical forms.
During training, we learn a vector θ which maximizes our regularized objective function. In Pasupat and Liang (2015), the objective is defined as the log-likelihood over all consistent denotations y_i:

$$J_{ML}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(y_i \mid x_i, w_i) + \lambda\|\theta\|_1 \tag{3}$$

The gradient of this objective for a specific example is:

$$\nabla_{ML} = \sum_{z \in Z_i} q(z) \cdot R(z) \cdot \nabla \log p_\theta(y_i \mid x_i, w_i) \tag{4}$$

where $q(z) = p(z|x_i, w_i, R(z) = 1)$ and $R(z) \in [0, 1]$ is a reward indicating whether z is consistent.

In this work we compare two methods for tackling spuriousness: *Meritocratic Gradient* (Guu et al., 2017) and *top-ML* (Zhang et al., 2017). The Meritocratic Gradient is a variant of the maximum likelihood gradient, where $q(z)$ is defined as a uniform distribution over Z_i^+:

$$\nabla_{merit} = \sum_{z \in Z_i^+} \frac{1}{|Z_i^+|} \cdot \nabla \log p_\theta(y_i \mid x_i, w_i) \tag{5}$$

Since spurious formulas vary greatly in form, their contributions to gradient updates may cancel each other, while updates of correct formulas will all vote in the same direction.

The top-ML objective is a variant of the log-likelihood objective which increases the margin between consistent and inconsistent logical forms. We denote z_i^+ and z_i^- as the logical forms with the highest model probability from Z^+ and Z_i^- respectively. The objective function:

$$J_{top-ML}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log \frac{p^+}{p^-} + \lambda \|\theta\|_1 \tag{6}$$

where $p^+ = p(z_i^+|x_i, w_i)$ and $p^- = p(z_i^-|x_i, w_i)$. The gradient of this objective for a specific example:

$$\nabla_{top-ML} = \phi(z_i^+) - \phi(z_i^-) \tag{7}$$

Optimization is performed using AdaGrad (Duchi et al., 2011). At test time, the algorithm selects the logical form $z \in Z$ with the highest model probability and executes it on w to predict its denotation.

5.2 Similarity based learning

We insert our similarity rewards into the gradient update stages for both methods.

Similarity top-ML: Eliminating the least similar consistent logical forms, Z^+ defined as:

$$Z^+ = \{z_i \mid R_{elim}(q, Z^+)_i = 1\}$$

The rest of the gradient computation remains the same as the original *top-ML*.

Similarity Meritocratic Gradient: Weighting the gradients of consistent logical forms according to their similarity score. The gradient defined as:

$$\nabla_d = \sum_{z \in Z} R_{soft}(z) \cdot \nabla \log p_\theta(y_i \mid x_i, w_i) \tag{8}$$

6 Experiments

6.1 Evaluating Similarity Model

Our similarity-based rewards are meant to reduce spuriousness. To measure our model's success we manually annotated 410 examples from the dataset and evaluated its success in ranking correct formulas the highest. For each example, S_i is the set of spurious formulas and Z_i the set of all the consistent formulas. We define the *noise* of an example as:

$$\frac{\sum_{z \in S_i} R(z)}{\sum_{z \in Z_i} R(z)}$$

The total *data noise* is the average noise for all the examples. We compute noise for both our Bag of Word and LSTM similarity models. To each model, we apply both softmax and elimination methods for computing rewards. The elimination method was tested using several ratios for filtering less similar formulas, the higher the ratio the more we filter. Table 2 lists the *data noise* for each similarity and, in parentheses, the number of questions where the model failed to reward any correct formula. We note that a high elimination ratio will discard all correct formulas for some examples, thus negatively affecting the parser learning. Optimally a model should minimize the noise whilst keeping as many correct logical forms as possible.

For our initial data we define $R(z) = 1$ for each z, setting the initial data noise at 0.72. Results in Table 2 show the simple Bag of Words model achieves the lowest noise, outperforming the LSTM model. In addition we compute the noise of the data using the parser's top-ML objective as the reward, where $R(z) = 1$ iff z is the top ranked formula by the parser. With data noise of 0.45, it is slightly better than our BOW model.

	BOW	LSTM
Softmax	0.50 (117)	0.72 (114)
Elimination:		
- 0.5	0.67 (146)	0.71 (177)
- 0.9	0.55 (190)	0.69 (256)
- 0.99	**0.48** (198)	0.69 (281)

Table 2: Measuring our models noise. Softmax averages softmax weights on similarities; elimination measures noise after eliminating least similar forms.

6.2 Parser Experiments

We run our parser on the development sets (three random 80:20 splits of the training data) and test data. In both settings the tables we test on do not appear during training. We measure parser *accuracy*, defined as the number of examples (x, w, y) on which the parser outputs correct answer y.

We tested the parser with our methods for gradient update (Section 5), and conducted an additional experiment where the similarity score was used as a new *feature* of our parser. Given the results in Section 6.1, we chose the Bag of Words model as our similarity function.

Results in Table 3[2] clearly show the *top* objective outperforms the original ML objective (43.1% to 37.5%). Training the parser with the top objective on filtered training set using similarity based elimination, failed to increase overall accuracy (42.1%). Applying the meritocratic gradient updates (Guu et al., 2017) reduced the parser accuracy, though our similarity-based gradient updates have drastically improved this method (25.5% to 36.2%). Finally plugging our similarity model as a feature of the base parser had increased the overall accuracy (43.1% to 44.5%) surpassing the state of the art 43.7% accuracy of Zhang et al. (2017).

	Dev	Test
Pasupat & Liang (2015)	37.0%	37.1%
Haug et al. (2017)	-	38.7%
Zhang et al. (2017)	40.4%	43.7%
Ours:		
- base + ML objective	35.4%	37.5%
- base + top objective	40.0%	43.1%
- top + 0.99 elimination	38.7%	41.4%
- top + 0.6 elimination	39.7%	42.1%
- meritocratic	23.2%	25.5%
- sim-meritocratic	34.2%	36.2%
- similarity as feature	**40.8**%	**44.5**%

Table 3: Parser accuracy and comparison with previous work.

7 Discussion and Related Work

Our work tackles the problem of learning from spurious logical forms for semantic parsing on question-denotation pairs. We note the *meritocratic gradient updates* (Guu et al., 2017) and the *top-ML* objective (Zhang et al., 2017) are two opposing variants of the log-likelihood objective. Meritocratic gradient updates are less dependent on the model, while the top-ML objective strictly follows it. Our intuition for using meritocratic gradient updates was that since spurious formulas outnumber correct ones, the model is inherently biased towards them. Therefore, using a method less reliant on the model should help. However, parser results on WikiTables Questions achieved the lowest accuracy when meritocratic gradient updates were applied, performing much worse than the top-ML objective. In addition, the top-ML objective reduces the *data noise* to 0.45, slightly lower than our similarity model. A possible explanation is that while spurious formulas greatly outnumber correct ones, many of them are in fact quite similar to the correct formula. Given such similarity, their gradient updates parameters in a similar direction to the correct gradient. Still, our addition of *similarity based gradient updates* significantly improved the basic meritocratic gradient updates. This indicates similarity based updates might improve results on a different task where spurious formulas are more distinct, such as SCONE (Guu et al., 2017).

In addition, though trained on the PPDB corpus, our Bag of Words model is able to identify the correct logical form with nearly the same accuracy as the fully trained parser of Zhang et al. (2017) (Section 6.1). Following this success of the Bag of Word model, we add it as an additional feature to the base parser. This addition improves the accuracy to 44.5% surpassing the former state of the art accuracy of 43.7% (Zhang et al., 2017).

[2]base results slightly different from Zhang et al., 2017 due to canonical utterance grammar

Using paraphrasing models for semantic parsing had been previously used in Berant and Liang (2014). Haug et al. (2016) also convert logical forms to utterances used in computing a similarity score with the original question. They re-rank the logical forms based on similarity, though they do not address the issue of spurious formulas.

References

Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. 2014.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. Driving semantic parsing from the world's response. *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010)*, 2010.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. *Association for Computational Linguistics (ACL), 2017*, 2017.

Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. Neural multi-step reasoning for question answering on semi-structured tables. In *eprint arXiv:1702.06589*, 2016.

Percy Liang. Lambda dependency-based compositional semantics. *CoRR*, abs/1309.4408, 2013. URL `http://arxiv.org/abs/1309.4408`.

Percy Liang, Michael I. Jordan, and Dan Klein. Learning dependency-based compositional semantics. *Association for Computational Linguistics (ACL), 2011*, 2011.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. 2015.

Yushi Wang, Jonathan Berant, and Percy Liang. Building a semantic parser overnight. pages 1332–1342, 01 2015.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *ICLR*, abs/1511.08198, 2016.

Yuchen Zhang, Panupong Pasupat, and Percy Liang. Macro grammars and holistic triggering for efficient semantic parsing. *Empirical Methods on Natural Language Processing (EMNLP), 2017*, 2017.

Appendix : LSTM Model

Motivation

The main limitation of the Bag of Words model is that it disregards words order and context, therefore harming the learning of similarity between phrases. Word order and context prove important since short phrases of the question are often paraphrased in the canonical utterance to a fixed sequence of words. Consider the question *"Who is the player who **won** the race?"* mapped to the canonical utterance *"The player who has **the highest rank**"*. The word *"won"* matches the phrase *"the highest rank"*, but a bag of words model might not capture such similarity. Additionally, the BOW model is pre-trained on a different dataset (PPDB). This affects the model's learning of rare words since unknown words are handled by sampling a random embedding. We address these issues by training a **LSTM** model directly on the *WikiTable Questions* dataset. Our training set consists of questions, along with their *consistent* and *inconsistent* canonical utterances generated by a trained semantic parser. The LSTM model does train on spurious logical forms as some of the consistent canonical utterances are in fact spurious. By training with cosine similarity loss instead of the log-likelihood objective, we expected spurious canonical utterances to be less similar to the original utterance, s.t their embeddings will have a lower cosine similarity than correct ones, allowing the model to learn mostly from the correct utterances.

The Model[1]

We use a standard LSTM with peephole from Gers et al. (2002). The input to the LSTM is the sentence's word embeddings (which we also learn) and the LSTM outputs the sentence embedding. Formally, the equations are:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}W_w^{x_t} + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}W_w^{x_t} + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}W_w^{x_t} + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}W_w^{x_t} + W_{ho}h_{t-1} + W_{co}c_t + b_c) \\
h_t &= o_t \tanh(c_t) \\
g_{LSTM}(x) &= h_{-1}
\end{aligned}
\tag{9}
$$

[1]code for training and evaluation is available at https://github.com/ResslerAL/nlpProj

Here W_w is the word embedding matrix, which we also learn.

Training the LSTM

Training data, X, consists of triplets (x, z^+, z^-) where x is a question in natural language, z^+ is a consistent canonical utterance (i.e. its formula outputs the correct result) and z^- is an incorrect canonical utterance. The objective is defined as:

$$
\begin{aligned}
&\min_{W_w} \frac{1}{|X|} \sum_{(x_1,x_2)\in X} \max(0, \delta - cos(g(x), g(z^+)) \\
&+cos(g(x), g(z^-))) + \lambda_1 ||W_{LSTM}|| + \\
&\lambda_2 ||W_{winitial} - W_w||
\end{aligned}
\tag{10}
$$

where W_{LSTM} are the LSTM parameters. Two optimizations were made to the training procedure. The first was filtering the data using the Bag of Words model. We ran the bag of words on the dataset and for each question left only the top scored 0.4% canonical utterances according to that model.
The second optimization used in training was an exploration-exploitation method. In exploration, we uniformly sample an example from the dataset. In exploitation we uniformly sample a question and then select its consistent and inconsistent canonical utterances with the highest probability under the current model. We use probability α to determine the sampling strategy, allowing us to control the ratio of exploration-exploitation in our training.

Conclusions

As seen in our experiments (section 6), The LSTM model failed to reduce spuriousness (compared to the Bag of Words model). There are several explanations to this result: First, the LSTM is an extremely expressive model, therefore it overfits the training set without learning anything substantial. This issue might have been resolved if we would have used pre-trained embeddings to reduce the model's number of parameters. Second, in many cases the canonical utterances are not grammatical, so it might be difficult to learn features over this kind of input. Since the Bag of Words model had performed surprisingly well, we decided to work with it instead of improving the LSTM.