Learning from Execution for Semantic Parsing

Bailin Wang[†], Mirella Lapata[†] and Ivan Titov[†§]

[†] ILCC, University of Edinburgh, [§] ILLC, University of Amsterdam

Semantic Parsing: Introduction

Data Example

Domain: Restaurant
NL: list all 3 star rated thai restaurants
Program: SELECT restaurant WHERE star_rating = 3 AND cuisine = thai

Task:
Semantic parsing aims at mapping a natural language (**NL**) utterance to its corresponding executable program.

Motivation

Challenges of semantic parsing:

- Current neural seq2seq parsers are data-hungry.
- Annotation of NL-Program pairs is very expensive.
- We need to do annotation for each new domain.

Motivation

Challenges of semantic parsing:

- Current neural seq2seq parsers are data-hungry.
- Annotation of NL-Program pairs is very expensive.
- We need to do annotation for each new domain.

In this work, we focus on the semi-supervised setting.

- No annotations available for most utterances.
- This setting resembles a common real-life scenario .

Motivating Example for Semi-Supervised Learning

Example

NL: list all 3 star rated thai restaurants

Candidate Programs	Gold	Exe
SELECT restaurant WHERE star_rating = thai	X	X
SELECT restaurant WHERE cuisine > 3	X	X
SELECT restaurant WHERE star_rating = 3	X	✓
SELECT restaurant WHERE star_rating = 3 AND cuisine = thai	✓	✓

Key Observations:

- Not all candidate programs for an utterance make sense.
- Executability is a weak yet free learning signal.

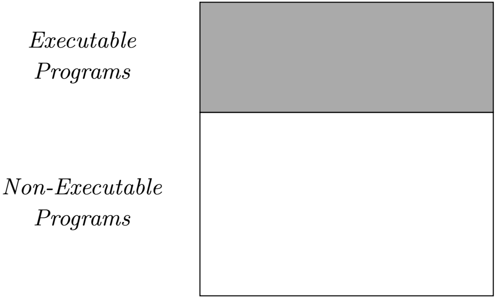
Ultilize Executability for Semi-Supervised Learning

Maximum marginal likelihood (MML):

$$\mathcal{L}_{\boldsymbol{\theta}}(x) = -\log \sum_{y} R(y) p(y|x, \boldsymbol{\theta})$$

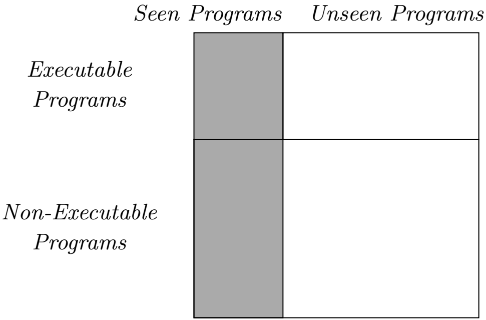
where x, y denote NL and program respectively. $R(y)$ returns 1 if y is executable; it returns 0 otherwise.

Challenge of MML Training: Large Search Space



Challenge:
The space of all possible programs is exponentially large, as well as the space of executable ones.

Explore by Beam-Search



Beam search:
It is typical to use beam search to explore the program space. As a result, the space can be further divided by whether a program is 'seen', i.e., retrieved by beam search.

Search Space Divided by Executability and Beam-Search

	Seen Programs	*Unseen Programs*
Executable Programs	P_{SE}	P_{UE}
Non-Executable Programs	P_{SN}	P_{UN}

Divided program space:
Beam-search can help us find a subset of executable programs (P_{SE}), but also ignores unseen executable programs (P_{UE}).

Two Conventional Approximations

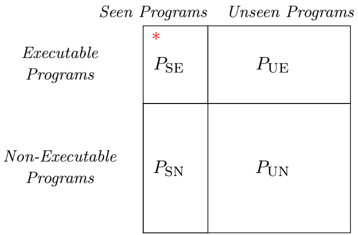


Figure: Divided program space. $*$ denotes the most probable executable program y^*.

1. **Self-Training**:

$$\mathcal{L}_{\mathsf{ST}}(x, \boldsymbol{\theta}) = -\log p(y^*|x, \boldsymbol{\theta})$$

2. **Top-K MML**:

$$\mathcal{L}_{\text{top-k}}(x, \boldsymbol{\theta}) = -\log \sum_{y \in P_{\text{SE}}} p(y|x, \boldsymbol{\theta})$$

Two Conventional Approximations

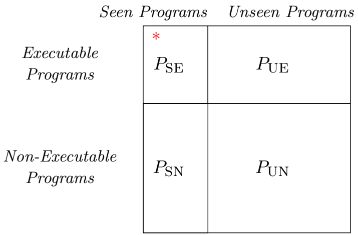


Figure: Divided program space. $*$ denotes the most probable executable program y^*.

1. **Self-Training**:

$$\mathcal{L}_{\mathsf{ST}}(x, \boldsymbol{\theta}) = -\log p(y^*|x, \boldsymbol{\theta})$$

2. **Top-K MML**:

$$\mathcal{L}_{\text{top-k}}(x, \boldsymbol{\theta}) = -\log \sum_{y \in P_{\mathrm{SE}}} p(y|x, \boldsymbol{\theta})$$

Can we design better objectives?

Motivations of Our New Objectives

	Seen Programs	*Unseen Programs*
Executable Programs	P_{SE}	P_{UE}
Non-Executable Programs	P_{SN}	P_{UN}

Figure: Divided program space.

- Encourage exploration of unseen executable programs.

Motivations of Our New Objectives

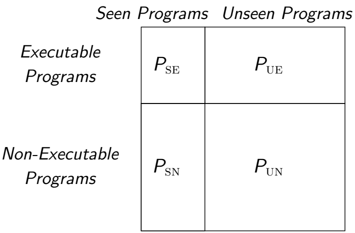


Figure: Divided program space.

- Encourage exploration of unseen executable programs.
- Promote sparsity among executable programs.

New Perspective of MML From Posterior Regularization

We assume a constrained faimily of distribution $\mathcal{Q}$: for any $\boldsymbol{q} \in \mathcal{Q}$,

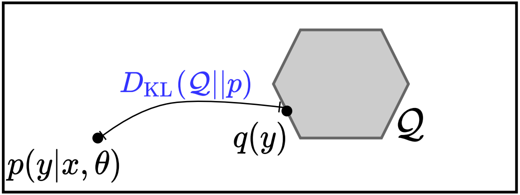
$$\mathbb{E}_{\boldsymbol{q}(y)}[R(y)] = 1$$

New Perspective of MML From Posterior Regularization

We assume a constrained faimily of distribution $\mathcal{Q}$: for any $\boldsymbol{q} \in \mathcal{Q}$,

$$\mathbb{E}_{\boldsymbol{q}(y)}[R(y)] = 1$$

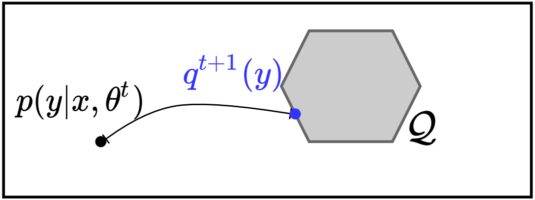
For a semantic parser $p(y|x, \boldsymbol{\theta})$, the objective of posterior regularization (Ganchev et al., 2010) is to penalize the KL-divergence between $\mathcal{Q}$ and p.



where $D_{\mathrm{KL}}(\mathcal{Q}||p) = \min_{q \in \mathcal{Q}} D_{\mathrm{KL}}[q(y)||p(y|x, \boldsymbol{\theta})]$.

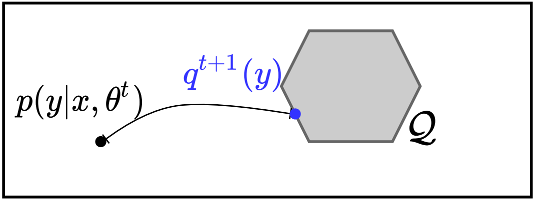
EM Algorithm for Optimizing PR

E-Step:

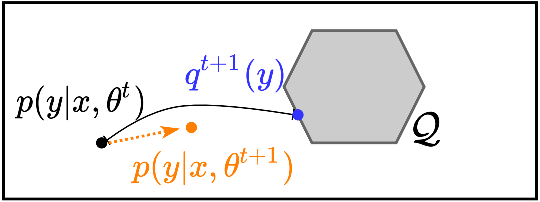


EM Algorithm for Optimizing PR

E-Step:



M-step:



E-Step Solution

	Seen Programs	*Unseen Programs*
Executable Programs	P_{SE}	P_{UE}
Non-Executable Programs	P_{SN}	P_{UN}

Figure: Divided program space.

E-step has a closed solution:

$$q^{t+1}(y) = \begin{cases} \frac{p(y|x,\boldsymbol{\theta}^t)}{p(P_{\text{SE}} \cup P_{\text{UE}})} & y \in P_{\text{SE}} \cup P_{\text{UE}} \\ 0 & \text{otherwise} \end{cases}$$

Intuitively, $q^{t+1}(y)$ is a renormalized version of p over executable programs.

Connect PR with MML

Self-Training and TopK-MML can be re-interpreted as two ways of finding $q^{t+1}(y)$ during E-step.

	Seen Programs	*Unseen Programs*
Executable Programs	* P_{SE}	P_{UE}
Non-Executable Programs	P_{SN}	P_{UN}

Self-Training:

$$q_{\text{ST}}^{t+1}(y) = \begin{cases} 1 & y = y^* \\ 0 & \text{otherwise} \end{cases}$$

Top-K MML:

$$q_{\text{top-k}}^{t+1}(y) = \begin{cases} \frac{p(y|x,\boldsymbol{\theta}^t)}{p(P_{\text{SE}})} & y \in P_{\text{SE}} \\ 0 & \text{otherwise} \end{cases}$$

Gradient Descent in M-Step

$q^{t+1}(y)$ as a 'pseudo label' for optimizing $p(y|x, \boldsymbol{\theta}^t)$.

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \nabla_{\boldsymbol{\theta}} \text{CrossEntropy}\big(q^{t+1}(y), p(y|x, \boldsymbol{\theta}^t)\big)$$

If we plug in the E-step solution, the gradient of the cross-entropy loss wrt. to θ is exactly the gradient of MML wrt. to θ !

Gradient Descent in M-Step

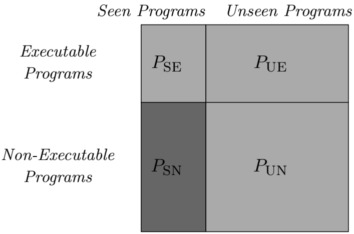
$q^{t+1}(y)$ as a 'pseudo label' for optimizing $p(y|x, \boldsymbol{\theta}^t)$.

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \nabla_{\boldsymbol{\theta}} \text{CrossEntropy}\big(q^{t+1}(y), p(y|x, \boldsymbol{\theta}^t)\big)$$

If we plug in the E-step solution, the gradient of the cross-entropy loss wrt. to θ is exactly the gradient of MML wrt. to θ !

optimize PR $\iff$ optimize MML

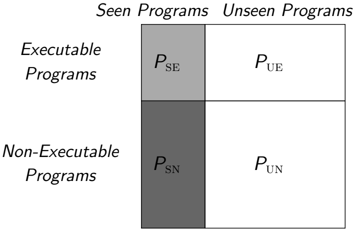
PR-based New Objective: Repulsion MML



$$q^{t+1}_{\text{repulsion}}(y) = \begin{cases} \frac{p(y|x,\boldsymbol{\theta}^t)}{1-p(P_{\mathrm{SN}})} & y \notin P_{\mathrm{SN}} \\ 0 & \text{otherwise} \end{cases}$$

Intuition: pushing away seen non-executable programs (P_{SN}); shift probability mass from the black area to the grey areas.

PR-based New Objective: Gentle MML



$$q^{t+1}_{\text{gentle}}(y) = \begin{cases} \frac{p(P_{\mathrm{SE}\cup\mathrm{SN}})}{p(P_{\mathrm{SE}})} p(y|x, \boldsymbol{\theta}^t) & y \in P_{\mathrm{SE}} \\ p(y|x, \boldsymbol{\theta}^t) & y \in P_{\mathrm{UE}} \cup P_{\mathrm{UN}} \\ 0 & y \in P_{\mathrm{SN}} \end{cases}$$

Intuition: it shifts the probability mass of seen non-executable programs (P_{SN}) directly to seen executable programs (P_{SE}) .

PR-based New Objective: Sparse MML

	Seen Programs	*Unseen Programs*
Executable Programs	P_{SE}	P_{UE}
Non-Executable Programs	P_{SN}	P_{UN}

$$q_{\text{sparse}}^{t+1} = \text{SparseMax}_{y \in P_{\mathrm{SE}}} \left(\log p(y|x, \boldsymbol{\theta}^t) \right)$$

Intuition: in most cases there is only one or few correct programs among all executable programs. (Also related to the low-density separation principle.)

Experiments

Overnight dataset:

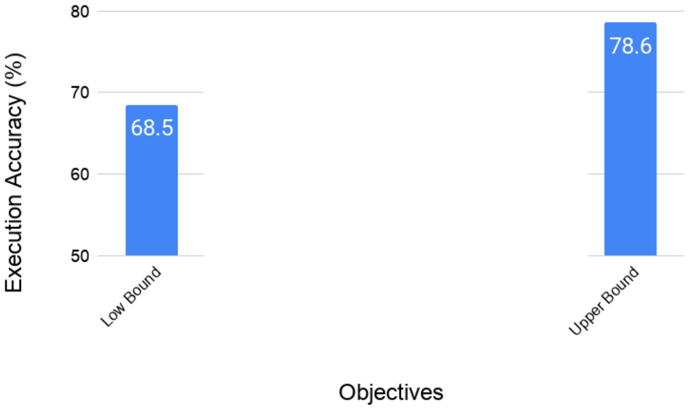
- It has eight different domains, each with labeled data

	Basketball	Blocks	Calendar	Housing	Publications	Recipes	Restaurants	Social
all	1952	1995	837	941	801	1080	1657	4419

Table: Number of data in each domain.

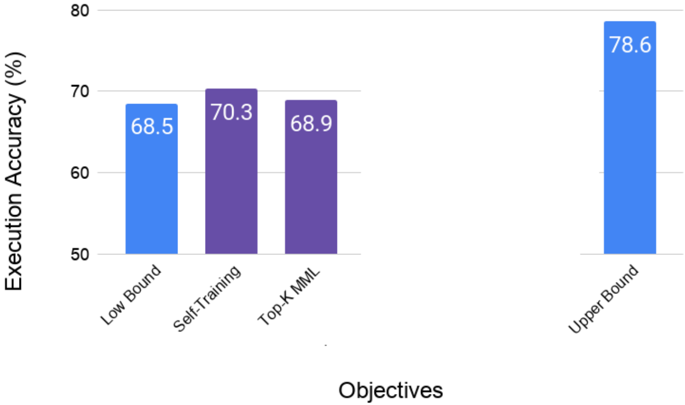
- For each domain, we simulate semi-supervised learning by sampling 30% data as labeled data and using the rest as unlabeled data.

Results of Lower and Upper Bound



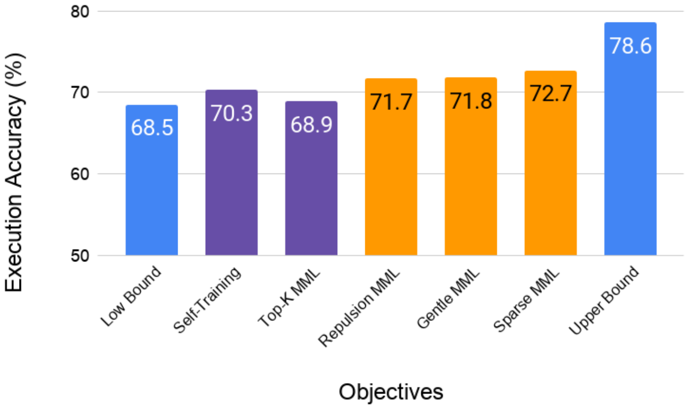
- There is a large gap between lower and upper bound.

Results of Baselines



- Self-Training and Top-K MML perform better than the lower bound, but the gap is still large.

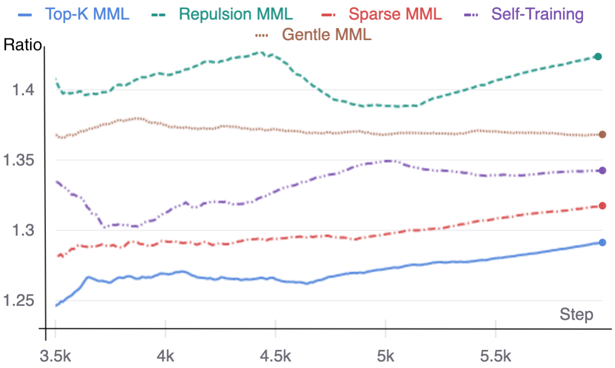
Results of Our New Objectives



- In average accuracy, Sparse MML achieves the best performance.

Analysis: Length Ratio

Length Ratio: length of programs (y) / length of utterances (x)



- Top-K MML favors shorter programs.
- Repulsion MML and Gentle MML prefer longer programs.
- Sparse MML strikes a balance between ST and Top-K MML.

Key Takeaways

- Executability can be used as weak learning signals for semi-supervised semantic parsing.
- Maximum marginal likelihood (MML) has a new interpretion from the perspective of posterior regularization.
- Our new objectives derived from the PR perspective can achieve better performance than Self-Training and TopK-MML.
- Code available at `http://github.com/berlino/tensor2struct-public`.

Thank you.

References I

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049.