Jointly Learning Semantic Parser and Natural Language Generator via Dual Information Maximization

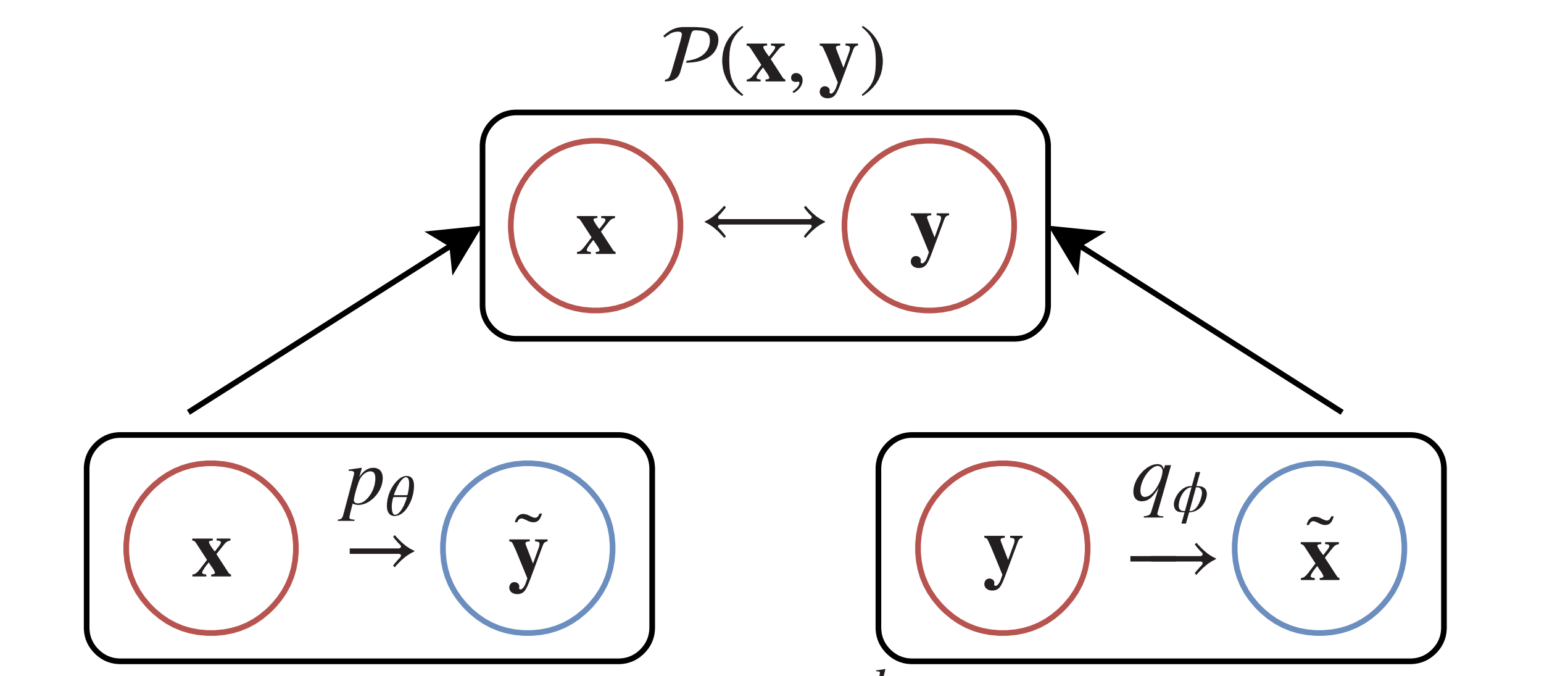
• Hai Ye, Information Science Institute, USC, US, hye@usc.edu • Wenjie Li, The Hong Kong Polytechnic University, HK, cswjli@comp.polyu.edu.hk • Lu Wang, Northeastern University, Boston, US, luwang@ccs.neu.edu

Problem Definition

Semantic Parser and NL Generator: Semantic parser ($p_\theta(\mathbf{y}|\mathbf{x})$) is to map the natural language (NL) utterances $\mathbf{x}$ to the structured MRs $\mathbf{y}$, and NL generator ($q_\phi(\mathbf{x}|\mathbf{y})$) generates NL from MRs. θ and ϕ are model parameters for learning. We use seq2seq models for parser and generator and more details can be found in our paper.

DATA		EXAMPLE	Ave. Token
ATIS	x	can you list all flights from chicago to milwaukee	10.6
	y	(_lambda $0 e (_and (_flight $0) (_from $0 chicago: _ci) (_to $0 milwaukee: _ci)))	26.5
DJANGO	x	convert max_entries into a string, substitute it for self._max_entries.	11.9
	y	self._max_entries = int(max_entries)	8.2
CONALA	x	more pythonic alternative for getting a value in range not using min and max	9.7
	y	a = 1 if x < 1 else 10 if x > 10 else x	14.1

Jointly Learning Parser and Generator: We propose to *jointly model semantic parsing and NL generation by exploiting the interaction between the two tasks*. We leverage the joint distribution $\mathcal{P}(\mathbf{x},\mathbf{y})$ of NL and MR to represent the duality. The joint distributions of $p^e(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p_\theta(\mathbf{y}|\mathbf{x})$, which is estimated from semantic parser, and $p^d(\mathbf{x},\mathbf{y}) = q(\mathbf{y})q_\phi(\mathbf{x}|\mathbf{y})$, which is modeled by NL generator, are both expected to approximate $\mathcal{P}(\mathbf{x},\mathbf{y})$, the unknown joint distribution of NL and MR. $p(\mathbf{x})$ and $q(\mathbf{y})$ are marginal distributions which are estimated by language models.


$\mathcal{P}(\mathbf{x},\mathbf{y})$
x ⟷ y
x $\xrightarrow{p_\theta}$ ỹ
y $\xrightarrow{q_\phi}$ x̃

$p^e(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p_\theta(\mathbf{y}|\mathbf{x})$ $\quad p^d(\mathbf{x},\mathbf{y}) = q(\mathbf{y})q_\phi(\mathbf{x}|\mathbf{y})$

To achieve this, we introduce **dual information maximization (DIM)** to empirically optimize the variational lower bounds of both $p^e(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p_\theta(\mathbf{y}|\mathbf{x})$ and $p^d(\mathbf{x},\mathbf{y}) = q(\mathbf{y})q_\phi(\mathbf{x}|\mathbf{y})$.

Resources

Our paper and code:

Dual Information Maximization

Dual Information: With $p_\theta(\mathbf{y}|\mathbf{x})$ for the parser and $q_\phi(\mathbf{x}|\mathbf{y})$ for the generator, the joint distributions of $p^e(\mathbf{x},\mathbf{y})$ and $p^d(\mathbf{x},\mathbf{y})$ can be estimated as $p^e(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p_\theta(\mathbf{y}|\mathbf{x})$ and $p^d(\mathbf{x},\mathbf{y}) = q(\mathbf{y})q_\phi(\mathbf{x}|\mathbf{y})$, where $p(\mathbf{x})$ and $q(\mathbf{y})$ are marginals. The dual information $I_{p^{e,d}(\mathbf{x},\mathbf{y})}$ between the two distributions is defined as follows:

$$I_{p^{e,d}(\mathbf{x},\mathbf{y})} = I_{p^e(\mathbf{x},\mathbf{y})} + I_{p^d(\mathbf{x},\mathbf{y})} \triangleq$$
$$\mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log p^e(\mathbf{x},\mathbf{y}) + \mathbb{E}_{p^d(\mathbf{x},\mathbf{y})} \log p^d(\mathbf{x},\mathbf{y})$$

which is the combination of the two joint distribution expectations.

Maximizing Dual Information:

- **Lower bounds of dual information.** Taking $I_{p^e(\mathbf{x},\mathbf{y})}$ for example, we adopt variational approximation to deduce its lower bound and instead maximize the lower bound:

$$\begin{aligned}\mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log p^e(\mathbf{x},\mathbf{y}) &= \mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log p^e(\mathbf{x}|\mathbf{y})p^e(\mathbf{y}) \\ &= \mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log q_\phi(\mathbf{x}|\mathbf{y}) + \mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log q(\mathbf{y}) \\ +\mathbb{E}_{p^e(\mathbf{y})}\big[\mathrm{KL}(p^e(\mathbf{x}|\mathbf{y}) \| q_\phi(\mathbf{x}|\mathbf{y}))\big] &+ \mathbb{E}_{p^e(\mathbf{x}|\mathbf{y})}\big[\mathrm{KL}(p^e(\mathbf{y}) \| q(\mathbf{y}))\big] \\ &\geqslant \mathbb{E}_{p^e(\mathbf{x},\mathbf{y})}\big[\log q_\phi(\mathbf{x}|\mathbf{y}) + \log q(\mathbf{y})\big] = \mathcal{L}^e_{\mathrm{DIM}}(\theta,\phi)\end{aligned}$$

  where $\mathrm{KL}(\cdot\|\cdot)(\geqslant 0)$ is the Kullback-Leibler (KL) divergence.
  Therefore, to maximize $I_{p^e(\mathbf{x},\mathbf{y})}$, we can instead maximize its lower bound of $\mathcal{L}^e_{\mathrm{DIM}}$.
- We adopt Monte Carlo samples using the REINFORCE policy to approximate the gradient of $\mathcal{L}^e_{\mathrm{DIM}}(\theta,\phi)$ with regard to θ:

$$\begin{aligned}\nabla_\theta \mathcal{L}^e_{\mathrm{DIM}}(\theta,\phi) &= \mathbb{E}_{p_\theta(\mathbf{y}|\mathbf{x})} \nabla_\theta \log p_\theta(\mathbf{y}|\mathbf{x}) \cdot [\log q_\phi(\mathbf{x}|\mathbf{y}) + \log q(\mathbf{y}) - \mathbf{b}] \\ &= \mathbb{E}_{p_\theta(\mathbf{y}|\mathbf{x})} \nabla_\theta \log p_\theta(\mathbf{y}|\mathbf{x}) \cdot l(\mathbf{x},\mathbf{y};\phi) \\ &\approx \frac{1}{|\mathcal{S}|} \sum_{\hat{\mathbf{y}}_i \in \mathcal{S}} \nabla_\theta \log p_\theta(\hat{\mathbf{y}}_i|\mathbf{x}) \cdot l(\mathbf{x},\hat{\mathbf{y}}_i;\phi)\end{aligned}$$

  We adopt the baseline function $\mathbf{b}$ by empirically averaging the signals to stabilize the learning process. With prior $p_\theta(\cdot|\mathbf{x})$, we use beam search to generate a pool of MR candidates ($\mathbf{y}$), denoted as $\mathcal{S}$, for the input of $\mathbf{x}$.
  The gradient with regard to ϕ is then calculated as:

$$\nabla_\phi \mathcal{L}^e_{\mathrm{DIM}}(\theta,\phi) = \mathbb{E}_{p_\theta(\mathbf{y}|\mathbf{x})} \nabla_\phi \log q_\phi(\mathbf{x}|\mathbf{y}) \approx \frac{1}{|\mathcal{S}|} \sum_{\hat{\mathbf{y}}_i \in \mathcal{S}} \nabla_\phi \log q_\phi(\mathbf{x}|\hat{\mathbf{y}}_i)$$

Semi-supervised DIM (SEMIDIM): We extend DIM to leverage unlabeled data. We denote the unlabeled NL dataset as $\mathbb{U}_\mathbf{x} = \{\mathbf{x}_i\}$ and the unlabeled MR dataset as $\mathbb{U}_\mathbf{y} = \{\mathbf{y}_i\}$. To leverage $\mathbb{U}_\mathbf{x}$, we maximize the unlabeled objective $\mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x}} \log p(\mathbf{x})$. Our goal is to involve model parameters in the optimization process of $\mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x}} \log p(\mathbf{x})$, so that the unlabeled data can facilitate parameter learning.

- **Lower Bounds of Unsupervised Objective.** The lower bound of $\mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x}} \log p(\mathbf{x})$ is as follows:

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x}} \log p(\mathbf{x}) &\geq \mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x},\mathbf{y}\sim p_\theta(\cdot|\mathbf{x})} \log p(\mathbf{x})p_\theta(\mathbf{y}|\mathbf{x}) \\ &\geq \mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x},\mathbf{y}\sim p_\theta(\cdot|\mathbf{x})}\big[\log q_\phi(\mathbf{x}|\mathbf{y}) + q(\mathbf{y})\big]\end{aligned}$$

Experiments

Datasets:

DATA	Train	Valid	Test	All
ATIS	4,480	480	450	5,410
DJANGO	16,000	1,000	1,805	18,805
CONALA	90,000	5,000	5,000	100,000

Baselines:
We compare DIM and SEMIDIM with the following baselines:

- **SUPER:** Train the parser or generator separately without joint learning. The models for the parser and generator are the same as DIM.
- **SELFTRAIN:** We use the pre-trained parser or generator to generate pseudo labels for the unlabeled sources, then the constructed pseudo samples will be mixed with the labeled data to fine-tune the pre-trained parser or generator.
- **BACKBOOST:** it generates sources from unlabeled targets. The training process for BACKBOOST is the same as in SELFTRAIN.

In addition to the above baselines, we also compare with popular supervised methods for each task, shown in the corresponding result tables.

Evaluation Metrics: Accuracy (*Acc.*) is reported for parser evaluation based on exact match, and BLEU-4 is adopted for generator evaluation. For the code generation task in CONALA, we use BLEU-4.

Main Results with Full- and Semi-supervision:
(We only report results on DJANGO. More results can be found in our paper.)

- Code generation and code summarization results on DJANGO. **Pro.**: proportion of the training samples used for training. Best result in each row is highlighted in bold. |full| = 16,000. Unlabeled code snippets are used for semi-supervised learning.

CODE GENERATION (in **Acc.**)				
Pro.	SUPER	DIM	SEMIDIM	BACKBOOST
1/8	42.3	44.9	**47.2**	47.0
1/4	50.2	51.1	**54.5**	51.7
3/8	52.2	53.7	54.6	**55.3**
1/2	56.3	58.4	**59.2**	58.9
full	65.1	**66.6**	–	–
Previous Supervised Methods (Pro. = full)				**Acc.**
LPN (Ling et al., 2016)				62.3
SNM (Yin and Neubig, 2017)				71.6
COARSE2FINE (Dong and Lapata, 2018)				74.1

CODE SUMMARIZATION (in **BLEU**)				
Pro.	SUPER	DIM	SEMIDIM	SELFTRAIN
1/8	54.1	56.0	**58.5**	54.4
1/4	57.1	61.4	**62.7**	58.0
3/8	63.0	64.3	**64.6**	63.0
1/2	65.2	66.3	**66.7**	65.4
full	68.1	**70.8**	–	–
Previous Supervised Methods (Pro. = full)				**BLEU**
DEEPCOM (Hu et al., 2018)				65.9

We first note the consistent advantage of DIM over SUPER across all proportions of training samples for learning. For semi-supervised scenarios, SEMIDIM delivers stronger performance than DIM, which only uses labeled data. Moreover, SEMIDIM outperforms both SELFTRAIN and BACKBOOST.