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

Hai Ye[♭]   **Wenjie Li**[♮]   **Lu Wang**[♯]

[♭] Information Sciences Institute, University of Southern California
[♮] Department of Computing, The Hong Kong Polytechnic University
[♯] College of Computer and Information Science, Northeastern University
hye@usc.edu, cswjli@comp.polyu.edu.hk, luwang@ccs.neu.edu

Abstract

Semantic parsing aims to transform natural language (NL) utterances into formal meaning representations (MRs), whereas an NL generator achieves the reverse: producing a NL description for some given MRs. Despite this intrinsic connection, the two tasks are often studied separately in prior work. In this paper, we model the *duality* of these two tasks via a joint learning framework, and demonstrate its effectiveness of boosting the performance on both tasks. Concretely, we propose the method of *dual information maximization* (DIM) to regularize the learning process, where DIM empirically maximizes the *variational lower bounds* of expected joint distributions of NL and MRs. We further extend DIM to a semi-supervision setup (SEMIDIM), which leverages unlabeled data of both tasks. Experiments on three datasets of dialogue management and code generation (and summarization) show that performance on both semantic parsing and NL generation can be consistently improved by DIM, in both supervised and semi-supervised setups[1].

1 Introduction

Semantic parsing studies the task of translating natural language (NL) utterances into formal meaning representations (MRs) (Zelle and Mooney, 1996; Tang and Mooney, 2000). NL generation models can be designed to learn the reverse: mapping MRs to their NL descriptions (Wong and Mooney, 2007). Generally speaking, MR often takes a logical form that captures the semantic meaning, including λ-calculus (Zettlemoyer and Collins, 2005, 2007), Abstract Meaning Representation (AMR) (Banarescu et al., 2013; Misra and Artzi, 2016), and general-purpose computer programs, such as

---

[1] Code for this paper is available at: https://github.com/oceanypt/DIM

$$p^e(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p_\theta(\mathbf{y}|\mathbf{x}) \quad p^d(\mathbf{x}, \mathbf{y}) = q(\mathbf{y})q_\phi(\mathbf{x}|\mathbf{y})$$
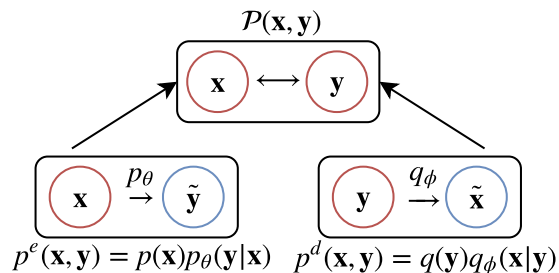
Figure 1: Illustration of our joint learning model. $\mathbf{x}$: NL; $\mathbf{y}$: MRs. $p_\theta(\mathbf{y}|\mathbf{x})$: semantic parser; $q_\phi(\mathbf{x}|\mathbf{y})$: NL generator. We model the duality of the two tasks by matching the joint distributions of $p^e(\mathbf{x}, \mathbf{y})$ (learned from semantic parser) and $p^d(\mathbf{x}, \mathbf{y})$ (learned from NL generator) to an underlying unknown distribution $\mathcal{P}(\mathbf{x}, \mathbf{y})$.

Python (Yin and Neubig, 2017) or SQL (Zhong et al., 2017). Recently, NL generation models have been proposed to automatically construct human-readable descriptions from MRs, for code summarization (Hu et al., 2018; Allamanis et al., 2016; Iyer et al., 2016) that predicts the function of code snippets, and for AMR-to-text generation (Song et al., 2018; Konstas et al., 2017; Flanigan et al., 2016).

Specifically, a common objective that semantic parsers aim to estimate is $p_\theta(\mathbf{y}|\mathbf{x})$, the conditional distribution between NL input $\mathbf{x}$ and the corresponding MR output $\mathbf{y}$, as demonstrated in Fig. 1. Similarly, for NL generation from MRs, the goal is to learn a generator of $q_\phi(\mathbf{x}|\mathbf{y})$. As demonstrated in Fig. 2, there is a clear *duality* between the two tasks, given that one task's input is the other task's output, and vice versa. However, such duality remains largely unstudied, even though joint modeling has been demonstrated effective in various NLP problems, e.g. question answering and generation (Tang et al., 2017), machine translation between paired languages (He et al., 2016), as well

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

Figure 2: Sample natural language utterances and meaning representations from datasets used in this work: Atis for dialogue management; Django (Oda et al., 2015) and ConaLa (Yin et al., 2018a) for code generation and summarization.

as sentiment prediction and subjective text generation (Xia et al., 2017).

In this paper, we propose to *jointly model semantic parsing and NL generation by exploiting the interaction between the two tasks*. Following previous work on dual learning (Xia et al., 2017), we leverage the joint distribution $\mathcal{P}(\mathbf{x}, \mathbf{y})$ of NL and MR to represent the duality. Intuitively, as shown in Fig. 1, 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.

To achieve this goal, we propose *dual information maximization* (DIM) (§3) to empirically optimize the *variational lower bounds* of the expected joint distributions of $p^e(\mathbf{x}, \mathbf{y})$ and $p^d(\mathbf{x}, \mathbf{y})$. Concretely, the coupling of the two expected distributions is designed to capture the dual information, with both optimized via variational approximation (Barber and Agakov, 2003) inspired by Zhang et al. (2018). Furthermore, combined with the supervised learning objectives of semantic parsing and NL generation, DIM bridges the two tasks within one joint learning framework by serving as a regularization term (§2.2). Finally, we extend supervised DIM to semi-supervision setup (SemiDIM), where unsupervised learning objectives based on unlabeled data are also optimized (§3.3).

We experiment with three datasets from two different domains: Atis for dialogue management; Django and ConaLa for code generation and summarization. Experimental results show that both the semantic parser and generator can be consistently improved with joint learning using DIM and SemiDIM, compared to competitive comparison models trained for each task separately.

Overall, we have the following contributions in

this work:

- We are the first to jointly study semantic parsing and natural language generation by exploiting the *duality* between the two tasks;
- We propose DIM to capture the duality and adopt variational approximation to maximize the dual information;
- We further extend supervised DIM to semi-supervised setup (SemiDIM).

2 Problem Formulation

2.1 Semantic Parsing and NL Generation

Formally, the task of semantic parsing is to map the input of NL utterances $\mathbf{x}$ to the output of structured MRs $\mathbf{y}$, and NL generation learns to generate NL from MRs.

Learning Objective. Given a labeled dataset $\mathbb{L} = \{\langle \mathbf{x}_i, \mathbf{y}_i \rangle\}$, we aim to learn a semantic parser ($\mathbf{x} \to \mathbf{y}$) by estimating the conditional distribution $p_\theta(\mathbf{y}|\mathbf{x})$, parameterized by θ, and an NL generator ($\mathbf{y} \to \mathbf{x}$) by modeling $q_\phi(\mathbf{x}|\mathbf{y})$, parameterized by ϕ. The learning objective for each task is shown below:

$$\mathcal{L}_{\text{parser}} = \mathbb{E}_{\langle \mathbf{x}, \mathbf{y} \rangle}[\log p_\theta(\mathbf{y}|\mathbf{x})] \quad (1)$$

$$\mathcal{L}_{\text{gen.}} = \mathbb{E}_{\langle \mathbf{x}, \mathbf{y} \rangle}[\log q_\phi(\mathbf{x}|\mathbf{y})] \quad (2)$$

Frameworks. Sequence-to-sequence (seq2seq) models have achieved competitive results on both semantic parsing and generation (Dong and Lapata, 2016; Hu et al., 2018), and without loss of generality, we adopt it as the basic framework for both tasks in this work. Specifically, for both $p_\theta(\mathbf{y}|\mathbf{x})$ and $q_\phi(\mathbf{x}|\mathbf{y})$, we use a two-layer bidirectional LSTM (bi-LSTM) as the encoder and another one-layer LSTM as the decoder with attention mechanism (Luong et al., 2015). Furthermore, we leverage pointer network (Vinyals et al., 2015) to copy tokens from the input to handle out-of-vocabulary (OOV) words. The structured MRs

are linearized for the sequential encoder and decoder. More details of the parser and the generator can be found in Appendix A. Briefly speaking, our models differ from existing work as follows:

PARSER: Our architecture is similar to the one proposed in Jia and Liang (2016) for semantic parsing;

GENERATOR: Our model improves upon the DEEPCOM coder summarization system (Hu et al., 2018) by: 1) replacing LSTM with bi-LSTM for the encoder to better model context, and 2) adding copying mechanism.

2.2 Jointly Learning Parser and Generator

Our joint learning framework is designed to model the duality between a parser and a generator. To incorporate the duality into our learning process, we design the framework to encourage the expected joint distributions $p^e(\mathbf{x}, \mathbf{y})$ and $p^d(\mathbf{x}, \mathbf{y})$ to both approximate the unknown joint distribution of $\mathbf{x}$ and $\mathbf{y}$ (shown in Fig. 1). To achieve this, we introduce *dual information maximization* (DIM) to empirically optimize the *variational lower bounds* of both $p^e(\mathbf{x}, \mathbf{y})$ and $p^d(\mathbf{x}, \mathbf{y})$, in which the coupling of expected distributions is captured as dual information (detailed in §3.1) and will be maximized during learning.

Our joint learning objective takes the form of:

$$\max_{\theta,\phi} \mathcal{L}(\theta, \phi) = \mathcal{L}_{\text{parser}} + \mathcal{L}_{\text{gen.}} + \lambda \cdot \mathcal{L}_{\text{DIM}}(\theta, \phi) \quad (3)$$

$\mathcal{L}_{\text{DIM}}$ is the variational lower bound of the two expected joint distributions, specifically,

$$\mathcal{L}_{\text{DIM}} = \mathcal{L}_{\text{DIM}}^e \text{ (Eq. 6)} + \mathcal{L}_{\text{DIM}}^d \text{ (Eq. 9)} \quad (4)$$

where $\mathcal{L}_{\text{DIM}}^e$ and $\mathcal{L}_{\text{DIM}}^d$ are the lower bounds over $p^e(\mathbf{x}, \mathbf{y})$ and $p^d(\mathbf{x}, \mathbf{y})$ respectively. The hyperparameter λ trades off between supervised objectives and dual information learning. With the objective of Eq. 3, we jointly learn a parser and a generator, as well as maximize the dual information between the two. $\mathcal{L}_{\text{DIM}}$ serves as a regularization term to influence the learning process, whose detailed algorithm is described in §3.

Our method of DIM is model-independent. If the learning objectives for semantic parser and NL generator are subject to Eq. 1 and Eq. 2, we can always adopt DIM to conduct joint learning. Out of most commonly used seq2seq models for the parser and generator, more complex tree and graph structures have been adopted to model MRs (Dong and Lapata, 2016; Song et al., 2018). In this paper, without loss of generality, we study our joint-learning method on the widely-used seq2seq frameworks mentioned above (§2.1).

3 Dual Information Maximization

In this section, we first introduce dual information in §3.1, followed by its maximization (§3.2). §3.3 discusses its extension with semi-supervision.

3.1 Dual Information

As discussed above, we treat semantic parsing and NL generation as the dual tasks and exploit the duality between the two tasks for our joint learning. With conditional distributions $p_\theta(\mathbf{y}|\mathbf{x})$ 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:

$$I_{p^{e,d}(\mathbf{x},\mathbf{y})} = I_{p^e(\mathbf{x},\mathbf{y})} + I_{p^d(\mathbf{x},\mathbf{y})} \triangleq$$
$$\mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log p^e(\mathbf{x},\mathbf{y}) + \mathbb{E}_{p^d(\mathbf{x},\mathbf{y})} \log p^d(\mathbf{x},\mathbf{y}) \quad (5)$$

which is the combination of the two joint distribution expectations.

To leverage the duality between the two tasks, we aim to drive the learning of the model parameters θ and ϕ via optimizing $I_{p^{e,d}(\mathbf{x},\mathbf{y})}$, so that the expectations of joint distributions $p^e(\mathbf{x}, \mathbf{y})$ and $p^d(\mathbf{x}, \mathbf{y})$ will be both maximized and approximate the latent joint distribution $\mathcal{P}(\mathbf{x}, \mathbf{y})$, whose procedure is similar to the joint distribution matching (Gan et al., 2017). By exploiting the inherent probabilistic connection between the two distributions, we hypothesize that it would enhance the learning of both tasks on parsing $p_\theta(\mathbf{y}|\mathbf{x})$ and generation $q_\phi(\mathbf{x}|\mathbf{y})$. Besides, to approach the same distribution $\mathcal{P}(\mathbf{x}, \mathbf{y})$, the expected joint distributions can learn to be close to each other, making the dual models coupled.

3.2 Maximizing Dual Information

Here, we present the method for optimizing $I_{p^e(\mathbf{x},\mathbf{y})}$, which can also be applied to $I_{p^d(\mathbf{x},\mathbf{y})}$. In contrast to the parameter sharing techniques in most multi-task learning work (Collobert et al., 2011; Ando and Zhang, 2005), parameter θ for the parser and parameter ϕ for generator are independent in our framework. In order to jointly train the two models and bridge the learning of θ and ϕ, during the optimization of $I_{p^e(\mathbf{x},\mathbf{y})}$, where the parser is the primal model, we utilize the distributions of the dual task (i.e. the generator)
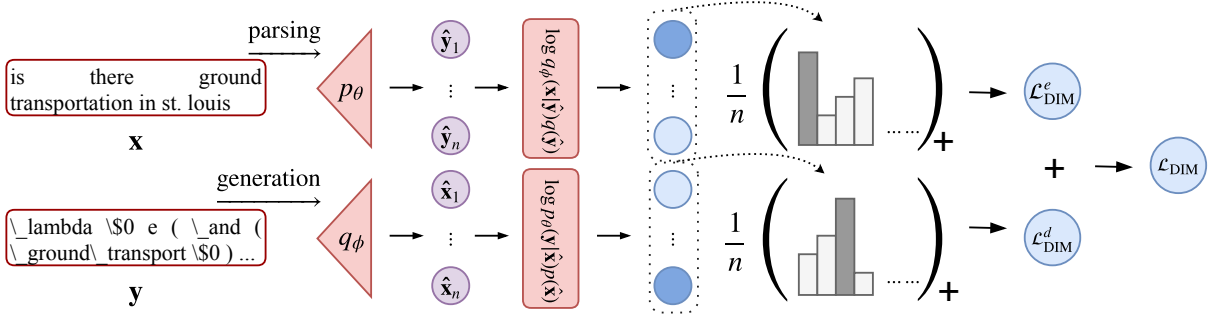
Figure 3: The pipeline of calculating lower bounds. We firstly use the parser or generator to sample MR or NL targets, then the sampled candidates go through the dual model and a language model to obtain the lower bounds.

to estimate $I_{p^e(\mathbf{x},\mathbf{y})}$. In this way, θ and ϕ can be both improved during the update of $I_{p^e(\mathbf{x},\mathbf{y})}$. Specifically, we rewrite $\mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log p^e(\mathbf{x}, \mathbf{y})$ as $\mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log p^e(\mathbf{y}) p^e(\mathbf{x}|\mathbf{y})$, where $p^e(\mathbf{y})$ and $p^e(\mathbf{x}|\mathbf{y})$ are referred as the dual task distributions. However, the direct optimization for this objective is impractical since both $p^e(\mathbf{y})$ and $p^e(\mathbf{x}|\mathbf{y})$ are unknown. Our solution is detailed below.

Lower Bounds of Dual Information. To provide a principled approach of optimizing $I_{p^e(\mathbf{x},\mathbf{y})}$, inspired by Zhang et al. (2018), we follow Barber and Agakov (2003) to adopt *variational approximation* to deduce its lower bound and instead maximize the lower bound. The lower bound deduction process is as following:

$$
\begin{aligned}
\mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log p^e(\mathbf{x}, \mathbf{y}) &= \mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log p^e(\mathbf{x}|\mathbf{y}) p^e(\mathbf{y}) \\
&= \mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log q_\phi(\mathbf{x}|\mathbf{y}) + \mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \log q(\mathbf{y}) \\
&\quad + \mathbb{E}_{p^e(\mathbf{y})} \big[\mathrm{KL}(p^e(\mathbf{x}|\mathbf{y}) \| q_\phi(\mathbf{x}|\mathbf{y})) \big] \\
&\quad + \mathbb{E}_{p^e(\mathbf{x}|\mathbf{y})} \big[\mathrm{KL}(p^e(\mathbf{y}) \| q(\mathbf{y})) \big] \\
&\geqslant \mathbb{E}_{p^e(\mathbf{x},\mathbf{y})} \big[\log q_\phi(\mathbf{x}|\mathbf{y}) + \log q(\mathbf{y}) \big] = \mathcal{L}^e_{\mathrm{DIM}}(\theta, \phi)
\end{aligned}
$$
(6)

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 $\mathcal{L}^e_{\mathrm{DIM}}$. $\mathcal{L}^e_{\mathrm{DIM}}$ is learned by using $q_\phi(\mathbf{x}|\mathbf{y})$ and $q(\mathbf{y})$ which approximate $p^e(\mathbf{x}|\mathbf{y})$ and $p^e(\mathbf{y})$. Besides, the lower bound of $\mathcal{L}^e_{\mathrm{DIM}}$ is the function of θ and ϕ, so in the process of learning $\mathcal{L}^e_{\mathrm{DIM}}$, the parser and generator can be both optimized.

As illustrated in Fig. 3, in the training process, to calculate the lower bound of $\mathcal{L}^e_{\mathrm{DIM}}$, we first use the being-trained parser to sample MR candidates for a given NL utterance. The sampled MRs then go through the generator and a marginal model (i.e., a language model of MRs) to obtain the final lower bound.

To learn the lower bound of $\mathcal{L}^e_{\mathrm{DIM}}$, we provide the following method to calculate its gradients:

Gradient Estimation. We adopt Monte Carlo samples using the REINFORCE policy (Williams, 1992) to approximate the gradient of $\mathcal{L}^e_{\mathrm{DIM}}(\theta, \phi)$ with regard to θ:

$$
\begin{aligned}
\nabla_\theta \mathcal{L}^e_{\mathrm{DIM}}(\theta, \phi) &= \mathbb{E}_{p_\theta(\mathbf{y}|\mathbf{x})} \nabla_\theta \log p_\theta(\mathbf{y}|\mathbf{x}) \\
&\quad \cdot [\log q_\phi(\mathbf{x}|\mathbf{y}) + \log q(\mathbf{y}) - \mathbf{b}] \\
&= \mathbb{E}_{p_\theta(\mathbf{y}|\mathbf{x})} \nabla_\theta \log p_\theta(\mathbf{y}|\mathbf{x}) \cdot l(\mathbf{x}, \mathbf{y}; \phi) \quad (7) \\
&\approx \frac{1}{|\mathcal{S}|} \sum_{\hat{\mathbf{y}}_i \in \mathcal{S}} \nabla_\theta \log p_\theta(\hat{\mathbf{y}}_i|\mathbf{x}) \cdot l(\mathbf{x}, \hat{\mathbf{y}}_i; \phi)
\end{aligned}
$$

$l(\mathbf{x}, \mathbf{y}; \phi)$ can be seen as the learning signal from the dual model, which is similar to the *reward* in reinforcement learning algorithms (Guu et al., 2017; Paulus et al., 2017). To handle the high-variance of learning signals, we adopt the baseline function $\mathbf{b}$ by empirically averaging the signals to stabilize the learning process (Williams, 1992). 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:

$$
\begin{aligned}
\nabla_\phi \mathcal{L}^e_{\mathrm{DIM}}(\theta, \phi) &= \mathbb{E}_{p_\theta(\mathbf{y}|\mathbf{x})} \nabla_\phi \log q_\phi(\mathbf{x}|\mathbf{y}) \\
&\approx \frac{1}{|\mathcal{S}|} \sum_{\hat{\mathbf{y}}_i \in \mathcal{S}} \nabla_\phi \log q_\phi(\mathbf{x}|\hat{\mathbf{y}}_i)
\end{aligned}
$$
(8)

The above maximization procedure for $\mathcal{L}^e_{\mathrm{DIM}}$ is analogous to the EM algorithm:

Step 1: Freeze ϕ and find the optimal $\theta^* = \arg\max_\theta \mathcal{L}^e_{\mathrm{DIM}}(\theta, \phi)$ with Eq. 7;

Step 2: Based on Eq. 8, with freezing θ^*, find the optimal $\phi^* = \arg\max_\phi \mathcal{L}^e_{\mathrm{DIM}}(\theta, \phi)$.

The two steps are repeated until convergence.

According to the gradient estimation in Eq. 7, when updating θ for the parser, we receive the *learning signal* $l(\mathbf{x}, \mathbf{y}; \phi)$ from the generator, and this learning signal can be seen as a *reward* from

the generator: if parser $p_\theta(\mathbf{y}|\mathbf{x})$ predicts high-quality MRs, the reward will be high; otherwise, the reward is low. This implies that the generator guides the parser to generate high-quality MRs, through which the lower bound for the expected joint distribution gets optimized. This also applies to the situation when we treat the generator as the primal model and the parser as the dual model.

The lower bound of $I_{p^d(\mathbf{x},\mathbf{y})}$ can be calculated in a similar way:

$$\mathbb{E}_{p^d(\mathbf{x},\mathbf{y})} \log p^d(\mathbf{x},\mathbf{y})$$
$$\geqslant \mathbb{E}_{p^d(\mathbf{x},\mathbf{y})}\big[\log p_\theta(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{x})\big] = \mathcal{L}^d_{\text{DIM}}(\theta, \phi) \quad (9)$$

which can be optimized the same way as in Eqs. 7 and 8 for estimating the gradients for $\mathcal{L}^d_{\text{DIM}}$.

Marginal Distributions. To obtain the marginal distributions $p(\mathbf{x})$ and $q(\mathbf{y})$, we separately train an LSTM-based language model (Mikolov et al., 2010) for NL and MR respectively, on each training set. Structured MRs are linearized into sequences for the sequential encoder and decoder in seq2seq models. Details on learning marginal distributions can be found in Appendix B.

Joint Learning Objective. Our final joint learning objective becomes:

$$\max_{\theta,\phi} \mathcal{J} = \sum_{\langle \mathbf{x},\mathbf{y}\rangle \in \mathbb{L}} \Big(\log p_\theta(\mathbf{y}|\mathbf{x}) + \log q_\phi(\mathbf{x}|\mathbf{y})$$
$$+ \lambda \sum_{\hat{\mathbf{y}}_i \sim p_\theta(\cdot|\mathbf{x})} \log q_\phi(\mathbf{x}|\hat{\mathbf{y}}_i) + \log q(\hat{\mathbf{y}}_i)$$
$$+ \lambda \sum_{\hat{\mathbf{x}}_i \sim q_\phi(\cdot|\mathbf{y})} \log p_\theta(\mathbf{y}|\hat{\mathbf{x}}_i) + \log p(\hat{\mathbf{x}}_i)\Big) \quad (10)$$

According to this learning objective, after picking up a data pair $\langle \mathbf{x},\mathbf{y}\rangle$, we will firstly calculate the supervised learning loss, then we sample MR candidates and NL samples using prior $p_\theta(\cdot|\mathbf{x})$ and $q_\phi(\cdot|\mathbf{y})$ respectively to obtain the corresponding lower bounds over $I_{p^e(\mathbf{x},\mathbf{y})}$ and $I_{p^d(\mathbf{x},\mathbf{y})}$.

3.3 Semi-supervised DIM (SEMIDIM)

We further extend DIM with semi-supervised learning. 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 leanring.

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, us-

ing the deduction in Ineq. 6:

$$\mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x}} \log p(\mathbf{x})$$
$$\geq \mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x},\mathbf{y}\sim p_\theta(\cdot|\mathbf{x})} \log p(\mathbf{x}) p_\theta(\mathbf{y}|\mathbf{x})$$
$$\geq \mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x},\mathbf{y}\sim p_\theta(\cdot|\mathbf{x})}\big[\log q_\phi(\mathbf{x}|\mathbf{y}) + q(\mathbf{y})\big] \quad (11)$$

Comparing Ineq. 11 to Ineq. 6, we can see that the unsupervised objective $\mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x}} \log p(\mathbf{x})$ and $I_{p^e(\mathbf{x},\mathbf{y})}$ share the same lower bound, so that the same optimization method from Eq. 7 and Eq. 8 can be utilized for learning the lower bound over $\mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x}} \log p(\mathbf{x})$.

Analysis. The lower bound of the unsupervised objective $\mathbb{E}_{\mathbf{x}\sim\mathbb{U}_\mathbf{x}} \log p(\mathbf{x})$ is a function of θ and ϕ. Therefore, updating this unsupervised objective will jointly optimize the parser and the generator. From the updating algorithm in Eq. 7, we can see that the parser $p_\theta(\mathbf{y}|\mathbf{x})$ is learned by using pseudo pair $(\mathbf{x},\hat{\mathbf{y}})$ where $\hat{\mathbf{y}}$ is sampled from $p_\theta(\cdot|\mathbf{x})$. This updating process resembles the popular semi-supervised learning algorithm of ***self-train*** that predicts pseudo labels for unlabeled data (Lee, 2013) and then attaches the predicted labels to the unlabeled data as additional training data. In our algorithm, the pseudo sample $(\mathbf{x},\hat{\mathbf{y}})$ will be weighted by the learning signal $l(\mathbf{x},\hat{\mathbf{y}};\phi)$, which decreases the impact of low-quality pseudo samples. Furthermore, from Eq. 8, the generator $q_\phi(\mathbf{x}|\mathbf{y})$ is updated using the pseudo sample $(\mathbf{x},\hat{\mathbf{y}})$, which is similar to the semi-supervised learning method of ***back-boost*** that is widely used in Neural Machine Translation for low-resource language pairs (Sennrich et al., 2016). Given the target-side corpus, back-boost generates the pseudo sources to construct pseudo samples, which is added for model training.

Similarly, to leverage the unlabeled data $\mathbb{U}_\mathbf{y}$ for semi-supervised learning, following Ineq. 11, we could also have the lower bound for $\mathbb{E}_{\mathbf{y}\sim\mathbb{U}_\mathbf{y}} \log p(\mathbf{y})$ as following,

$$\mathbb{E}_{\mathbf{y}\sim\mathbb{U}_\mathbf{y}} \log p(\mathbf{y})$$
$$\geq \mathbb{E}_{\mathbf{y}\sim\mathbb{U}_\mathbf{y},\mathbf{x}\sim q_\phi(\cdot|\mathbf{y})}\big[\log p_\theta(\mathbf{y}|\mathbf{x}) + p(\mathbf{x})\big] \quad (12)$$

which is the same as the lower bound of $I_{p^d(\mathbf{x},\mathbf{y})}$.

Semi-supervised Joint Learning Objective. From the above discussions, we can deduce the lower bounds for the unsupervised objectives to be the same as the lower bounds of the dual information. We thus have the following semi-supervised joint-learning objective:

$$\max_{\theta,\phi} \mathcal{J} = \sum_{\langle \mathbf{x},\mathbf{y}\rangle \in \mathbb{L}} \big(\log p_\theta(\mathbf{y}|\mathbf{x}) + \log q_\phi(\mathbf{x}|\mathbf{y})\big)$$

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

Table 1: Statistics of datasets used for evaluation. Around 500K additional samples of low confidence from CONALA are retained for model pre-training.

$$+\lambda \sum_{\mathbf{x}\sim\mathcal{D}_{\mathbf{x}},\hat{\mathbf{y}}_i\sim p_\theta(\cdot|\mathbf{x})} \big(\log q_\phi(\mathbf{x}|\hat{\mathbf{y}}_i) + \log q(\hat{\mathbf{y}}_i) \big)$$

$$+\lambda \sum_{\mathbf{y}\sim\mathcal{D}_{\mathbf{y}},\hat{\mathbf{x}}_i\sim q_\phi(\cdot|\mathbf{y})} \big(\log p_\theta(\mathbf{y}|\hat{\mathbf{x}}_i) + \log p(\hat{\mathbf{x}}_i) \big)$$

$$(13)$$

where $\mathcal{D}_{\mathbf{x}} = \mathbb{U}_{\mathbf{x}} \cup \mathbb{L}_{\mathbf{x}}$ and $\mathcal{D}_{\mathbf{y}} = \mathbb{U}_{\mathbf{y}} \cup \mathbb{L}_{\mathbf{y}}$. In this work, we weight the dual information and unsupervised objectives equally for simplicity, so the lower bounds over them are combined for joint optimization. We combine the labeled and unlabeled data to calculate the lower bounds to optimize the variational lower bounds of dual information and unsupervised objectives.

4 Experiments

4.1 Datasets

Experiments are conducted on three datasets with sample pairs shown in Fig. 2: one for dialogue management which studies semantic parsing and generation from λ-calculus (Zettlemoyer and Collins, 2007) (ATIS) and two for code generation and summarization (DJANGO, CONALA).

ATIS. This dataset has 5,410 pairs of queries (NL) from a flight booking system and corresponding λ-calculus representation (MRs). The anonymized version from Dong and Lapata (2016) is used.

DJANGO. It contains 18,805 lines of Python code snippets (Oda et al., 2015). Each snippet is annotated with a piece of human-written pseudo code. Similar to Yin and Neubig (2017), we replace strings separated by quotation marks with indexed *place_holder* in NLs and MRs.

CONALA. This is another Python-related corpus containing 598,237 intent/snippet pairs that are automatically mined from Stack Overflow (Yin et al., 2018a). Different from DJANGO, the intent in CONALA is mainly about the question on a specific topic instead of pseudo code. The full dataset contains noisy aligned pairs, and we keep the top 100,000 pairs of highest confidence scores for experiment and the rest for model pre-training.

For DJANGO and CONALA, the NL utterances

SEMANTIC PARSING (in **Acc.**)				
Pro.	SUPER	DIM	SEMIDIM	SELFTRAIN
1/4	64.7	69.0	**71.9**	66.3
1/2	78.1	78.8	**80.8**	79.2
full	84.6	**85.3**	–	–
Previous Supervised Methods (Pro. = full)				**Acc.**
SEQ2TREE (Dong and Lapata, 2016)				84.6
ASN (Rabinovich et al., 2017)				85.3
ASN+SUPATT (Rabinovich et al., 2017)				85.9
COARSE2FINE (Dong and Lapata, 2018)				87.7

NL GENERATION (in **BLEU**)				
Pro.	SUPER	DIM	SEMIDIM	BACKBOOST
1/4	36.9	37.7	39.1	**40.9**
1/2	39.1	40.7	**40.9**	39.3
full	39.3	**40.6**	–	–
Previous Supervised Methods (Pro. = full)				**BLEU**
DEEPCOM (Hu et al., 2018)				42.3

Table 2: Semantic parsing and NL generation results on ATIS. **Pro.**: proportion of the training samples used for training. Best result in each row is highlighted in bold. |full| = 4,434.

are lowercased and tokenized and the tokens in code snippets are separated with space. Statistics of the datasets are summarized in Table 1.

4.2 Experimental Setups

Joint-learning Setup. Before jointly learning the models, we pre-train the parser and the generator separately, using the labeled dataset, to enable the sampling of valid candidates with beam search when optimizing the lower bounds of dual information (Eqs. 7 and 8). The beam size is tuned from {3,5}. The parser and the generator are pre-trained until convergence. We also learn the language models for NL and MRs on the training sets beforehand, which are not updated during joint learning. Joint learning stops when the parser or the generator does not get improved for 5 continuous iterations. λ is set to 0.1 for all the experiments. Additional descriptions about our setup are provided in Appendix C.

For the semi-supervised setup, since ATIS and DJANGO do not have additional unlabeled corpus and it is hard to obtain in-domain NL utterances and MRs, we create a new partial training set from the original training set via subsampling, and the rest is used as the unlabeled corpus. For CONALA, we subsample data from the full training set to construct the new training set and unlabeled set instead of sampling from the low-quality corpus which will much boost the data volume.

Evaluation Metrics. Accuracy (*Acc.*) is reported

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	

Table 3: Code generation and code summarization results on DJANGO. |full| = 16,000.

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 following the setup in Yin et al. (2018a).

Baselines. We compare our methods of 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 (Lee, 2013)**:** 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: Adopted from the back translation method in Sennrich et al. (2016), which 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.

4.3 Results and Further Analysis

Main Results with Full- and Semi-supervision. Results on the three datasets with supervised and semi-supervised setups are presented in Tables 2, 3, and 4. For semi-supervised experiments on ATIS, we use the NL part as extra unlabeled sam-

CODE GENERATION (in **BLEU**)				
Pro.	SUPER	DIM	SEMIDIM	BACKBOOST
1/2	8.6	**9.6**	9.5	9.0
full	11.1	**12.4**	–	–

CODE SUMMARIZATION (in **BLEU**)				
Pro.	SUPER	DIM	SEMIDIM	SELFTRAIN
1/2	13.4	14.5	**15.1**	12.7
full	22.5	**24.8**	–	–
Previous Supervised Methods (Pro. = full)			**BLEU**	
CODE GEN.: NMT (Yin et al., 2018a)			10.7	
CODE SUM.: DEEPCOM (Hu et al., 2018)			20.1	

After Pre-training (in **BLEU**)				
	CODE GEN.		CODE SUM.	
Pro.	SUPER	DIM	SUPER	DIM
1/2	10.3	**10.6**	**23.1**	23.0
full	11.1	**12.4**	25.9	**26.3**
Previous Supervised Methods (Pro. = full)			**BLEU**	
CODE GEN.: NMT (Yin et al., 2018a)			10.9	
CODE SUM.: DEEPCOM (Hu et al., 2018)			26.5	

Table 4: Code generation and code summarization results on CONALA. For semi-supervised learning (Pro. = 1/2), we sample 30K code snippets from the left data (not used as training data) as unlabeled samples. |full| = 90,000.

ples following Yin et al. (2018b); for DJANGO and CONALA, unlabeled code snippets are utilized.

We first note the consistent advantage of DIM over SUPER across all datasets and proportions of training samples for learning. This indicates that DIM is able to exploit the interaction between the dual tasks, and further improves the performance on both semantic parsing and NL generation.

For semi-supervised scenarios, SEMIDIM, which employs unlabeled samples for learning, delivers stronger performance than DIM, which only uses labeled data. Moreover, SEMIDIM outperforms both SELFTRAIN and BACKBOOST, the two semi-supervised learning methods. This is attributed to SEMIDIM's strategy of re-weighing pseudo samples based on the learning signals, which are indicative of their qualities, whereas SELFTRAIN and BACKBOOST treat all pseudo samples equally during learning. Additionally, we study the pre-training effect on CONALA. As can be seen in Table 4, pre-training further improves the performance of SUPER and DIM on both code generation and summarization.

Model Analysis. Here we study whether DIM helps enhance the lower bounds of the expected joint distributions of NL and MRs. Specifically, lower bounds are calculated as in Eqs. 6 and 9 on the full training set for models of SUPER and
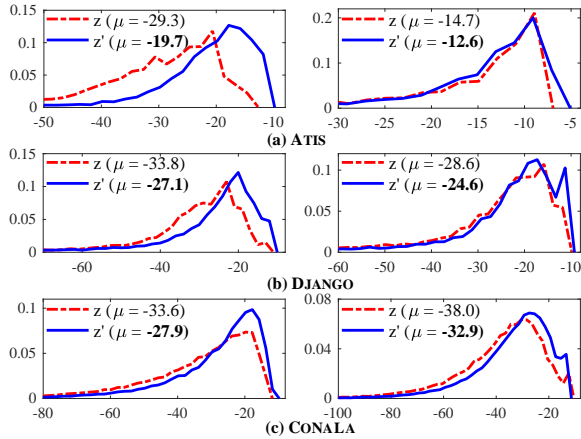
Figure 4: Lower bounds of the *full* training set. x-axis: lower bound value; y-axis: frequency. The left column is for semantic parsing, and the right column for NL generation. z is SUPER method and z' is DIM. μ is the average lower bound, with significantly better values boldfaced ($p < 0.01$).
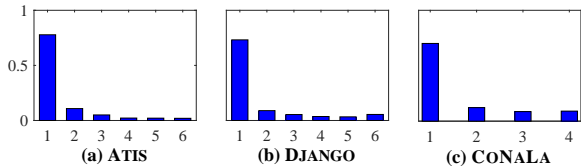

Figure 5: Distributions of the rank of learning signals over the gold-standard samples among the sampled set on unlabeled data using SEMIDIM (Pro. = 1/2).

PARSER	SUPER	🔒 GEN.	DIM
ATIS	84.6	84.2	85.3
DJANGO	65.1	65.8	66.6
CONALA	11.1	11.4	12.4
GENERATOR	SUPER	🔒 PARSER	DIM
ATIS	39.3	41.0	40.6
DJANGO	68.1	66.5	70.8
CONALA	22.5	23.0	24.8

Table 5: Ablation study with full training set by freezing (🔒) model parameters for generator or parser during learning. Darker indicates higher values.
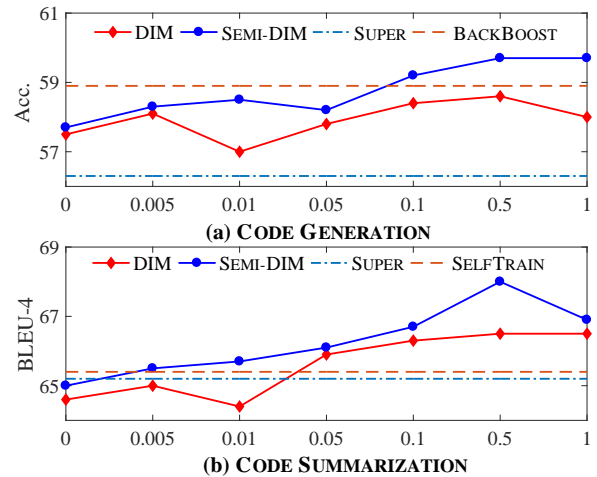

Figure 6: Model performance with different λ values on DJANGO (Pro. = 1/2).

DIM. As displayed in Fig. 4, DIM better optimizes the lower bounds of both the parser and the generator, with significantly higher values of average lower bounds on the full data. These results further explains that when the lower bound of the primal model is improved, it produces learning signals of high quality for the dual model, leading to better performance on both tasks.

As conjectured above, SEMIDIM outperforms SELFTRAIN in almost all setups because SEMIDIM re-weights the pseudo data with learning signals from the dual model. To demonstrate this, by giving the gold label for the unlabeled corpus, we rank the learning signal over the gold label among the sampled set using the semi-trained model, e.g. on ATIS, given an NL $\mathbf{x}$ from the dataset used as the unlabeled corpus, we consider the position of the learning signal $l(\mathbf{x}, \mathbf{y}^*; \phi)$ of gold-standard sample among all samples $\{l(\mathbf{x}, \hat{\mathbf{y}}_i; \phi) | \hat{\mathbf{y}}_i \in S\}$. As seen in Fig. 5, the gold candidates are almost always top-ranked, indicating that SEMIDIM is effective of separating pseudo samples of high and low-quality.

Ablation Study. We conduct ablation studies by training DIM with the parameters of parser or generator frozen. The results are presented in Table 5. As anticipated, for both of parsing and generation, when the dual model is frozen, the performance of the primal model degrades. This again demonstrates DIM's effectiveness of jointly optimizing both tasks. Intuitively, jointly updating both the primal and dual models allows a better learned dual model to provide high-quality learning signals, leading to an improved lower bound for the primal. As a result, freezing parameters of the dual model has a negative impact on the learning signal quality, which affects primal model learning.

Effect of λ. λ controls the tradeoff between learning dual information and the unsupervised learning objective. Fig. 6 shows that the optimal model performance can be obtained when λ is within $0.1 \sim 1$. When λ is set to 0, the joint training only employs labeled samples, and its performance decreases significantly. A minor drop is observed at $\lambda = 0.01$, which is considered to re-
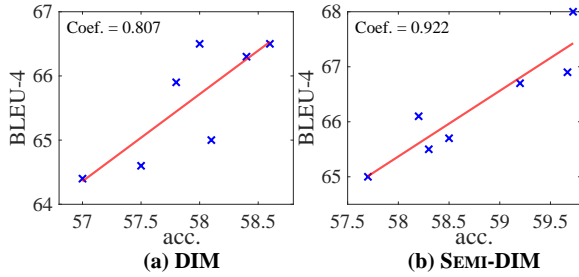
Figure 7: Performance correlation between parser and generator. x-axis is for parser and y-axis is for generator. Coef. indicates Pearson correlation coefficient.

sult from the variance of learning signals derived from the REINFORCE algorithm.

Correlation between Parser and Generator. We further study the performance correlation between the coupled parser and generator. Using the model outputs shown in Fig. 6, we run linear regressions of generator performance on parser performance, and a high correlation is observed between them (Fig. 7).

5 Related Work

Semantic Parsing and NL Generation. Neural sequence-to-sequence models have achieved promising results on semantic parsing (Dong and Lapata, 2016; Jia and Liang, 2016; Ling et al., 2016; Dong and Lapata, 2018) and natural language generation (Iyer et al., 2016; Konstas et al., 2017; Hu et al., 2018). To better model structured MRs, tree structures and more complicated graphs are explored for both parsing and generation (Dong and Lapata, 2016; Rabinovich et al., 2017; Yin and Neubig, 2017; Song et al., 2018; Cheng et al., 2017; Alon et al., 2018). Semi-supervised learning has been widely studied for semantic parsing (Yin et al., 2018b; Kociský et al., 2016; Jia and Liang, 2016). Similar to our work, Chen and Zhou (2018) and Allamanis et al. (2015) study code retrieval and code summarization jointly to enhance both tasks. Here, we focus on the more challenging task of code generation instead of retrieval, and we also aim for general-purpose MRs.

Joint Learning in NLP. There has been growing interests in leveraging related NLP problems to enhance primal tasks (Collobert et al., 2011; Peng et al., 2017; Liu et al., 2016), e.g. sequence tagging (Collobert et al., 2011), dependency parsing (Peng et al., 2017), discourse analysis (Liu et al., 2016). Among those, multi-task

learning (MTL) (Ando and Zhang, 2005) is a common method for joint learning, especially for neural networks where parameter sharing is utilized for representation learning. We follow the recent work on dual learning (Xia et al., 2017) to train dual tasks, where interactions can be employed to enhance both models. Dual learning has been successfully applied in NLP and computer vision problems, such as neural machine translation (He et al., 2016), question generation and answering (Tang et al., 2017), image-to-image translation (Yi et al., 2017; Zhu et al., 2017). Different from Xia et al. (2017) which minimizes the divergence between the two expected joint distributions, we aim to learn the expected distributions in a way similar to distribution matching (Gan et al., 2017). Furthermore, our method can be extended to semi-supervised scenario, prior to Xia et al. (2017)'s work which can only be applied in supervised setup. Following Zhang et al. (2018), we deduce the variational lower bounds of expected distributions via information maximization (Barber and Agakov, 2003). DIM aims to optimize the dual information instead of the mutual information studied in Zhang et al. (2018).

6 Conclusion

In this work, we propose to jointly train the semantic parser and NL generator by exploiting the structural connections between them. We introduce the method of DIM to exploit the duality, and provide a principled way to optimize the dual information. We further extend supervised DIM to semi-supervised scenario (SEMIDIM). Extensive experiments demonstrate the effectiveness of our proposed methods.

To overcome the issue of poor labeled corpus for semantic parsing, some automatically mined datasets have been proposed, e.g. CONALA (Yin et al., 2018a) and STAQC (Yao et al., 2018). However, these datasets are noisy and it is hard to train robust models out of them. In the future, we will further apply DIM to learn semantic parser and NL generator from the noisy datasets.

References

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *Proceedings of ICML*.

Miltiadis Allamanis, Daniel Tarlow, Andrew D. Gordon, and Yi Wei. 2015. Bimodal modelling of source code and natural language. In *Proceedings of ICML*.

Uri Alon, Omer Levy, and Eran Yahav. 2018. code2seq: Generating sequences from structured representations of code. *CoRR*, abs/1808.01400.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

David Barber and Felix V. Agakov. 2003. The IM algorithm: A variational approach to information maximization. In *Proceedings of NIPS*.

Qingying Chen and Minghui Zhou. 2018. A neural framework for retrieval and summarization of source code. In *Proceedings of ASE*.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of ACL*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of ACL*.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime G. Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of NAACL HLT*.

Zhe Gan, Liqun Chen, Weiyao Wang, Yunchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. 2017. Triangle generative adversarial networks. In *Proceedings of NIPS*.

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

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Proceedings of NIPS*.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018. Deep code comment generation. In *Proceedings of ICPC*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of ACL*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*.

Tomás Kociský, 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 EMNLP*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: sequence-to-sequence models for parsing and generation. In *Proceedings of ACL*.

Dong-Hyun Lee. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Fumin Wang, and Andrew W. Senior. 2016. Latent predictor networks for code generation. In *Proceedings of ACL*.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *Proceedings of AAAI*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*.

Dipendra Kumar Misra and Yoav Artzi. 2016. Neural shift-reduce CCG semantic parsing. In *Proceedings of EMNLP*.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation (T). In *Proceedings of ASE*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of ACL*.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of ACL*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. In *Proceedings of ACL*.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *CoRR*, abs/1706.02027.

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *Joint SIGDAT Conference on EMNLP and Very Large Corpora*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of NIPS*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

Yuk Wah Wong and Raymond J. Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of NAACL-HLT*.

Yingce Xia, Tao Qin, Wei Chen, Jiang Bian, Nenghai Yu, and Tie-Yan Liu. 2017. Dual supervised learning. In *Proceedings of ICML*.

Ziyu Yao, Daniel S. Weld, Wei-Peng Chen, and Huan Sun. 2018. Staqc: A systematically mined question-code dataset from stack overflow. In *Proceedings of WWW*.

Zili Yi, Hao (Richard) Zhang, Ping Tan, and Minglun Gong. 2017. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of ICCV*.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018a. Learning to mine aligned code and natural language pairs from stack overflow. In *Proceedings of MSR*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of ACL*.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018b. StructVAE: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of ACL*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of IAAI*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of EMNLP-CoNLL*.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. In *Proceedings of NIPS*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of ICCV*.

A Model Details for the Parser and Generator

The parser and generator have the same seq2seq framework. We take the parser for example. Given the NL utterance $\mathbf{x}$ and the linearized MR $\mathbf{y}$, we use bi-LSTM to encode $\mathbf{x}$ into context vectors, and then a LSTM decoder generates $\mathbf{y}$ from the context vectors. The parser $p_\theta(\mathbf{y}|\mathbf{x})$ is formulated as following:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p_\theta(y_t|\mathbf{y}_{<t}, \mathbf{x}) \tag{14}$$

where $\mathbf{y}_{<t} = y_1 \cdots y_{t-1}$.

The hidden state vector at time t from the encoder is the concatenation of forward hidden vector $\overrightarrow{\mathbf{h}}_t$ and backward one $\overleftarrow{\mathbf{h}}_t$, denoted as $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t]$. With the LSTM unit f_{LSTM_e} from the encoder, we have $\overrightarrow{\mathbf{h}}_t = f_{\text{LSTM}_e}(x_t, \overrightarrow{\mathbf{h}}_{t-1})$ and $\overleftarrow{\mathbf{h}}_t = f_{\text{LSTM}_e}(x_t, \overleftarrow{\mathbf{h}}_{t+1})$.

From the decoder side, using the decoder LSTM unit f_{LSTM_d}, we have the hidden state vector at time t as $\mathbf{s}_t = f_{\text{LSTM}_d}(y_{t-1}, \mathbf{s}_{t-1})$. Global attention mechanism (Luong et al., 2015) is applied to obtain the context vector $\mathbf{c}_t$ at time t:

$$\mathbf{c}_t = \sum_{i=1}^{|\mathbf{x}|} \alpha_{t,i} \mathbf{h}_i$$

where $\alpha_{t,i}$ is the attention weight and is specified as:

$$\alpha_{t,i} = \frac{\exp(\mathbf{W}_{\text{att}}[\mathbf{s}_t; \mathbf{h}_i])}{\sum_{k=1}^{|x|} \exp(\mathbf{W}_{\text{att}}[\mathbf{s}_t \mathbf{h}_k])} \tag{15}$$

where $\mathbf{W}_{\text{att}}$ is the learnable parameters.

At time t, with hidden state $\mathbf{s}_t$ in the decoder and context vector $\mathbf{c}_t$ from the encoder, we have the prediction probability for y_t:

$$p_{\text{vocab}}(y_t|\mathbf{y}_{<t}, \mathbf{x}) = \mathbf{f}_{\text{softmax}}(\mathbf{W}_{d_1} \cdot \tanh(\mathbf{W}_{d_2}[\mathbf{s}_t; \mathbf{c}_t]))$$

where $\mathbf{W}_{d_1}$ and $\mathbf{W}_{d_2}$ are learnable parameters.

We further apply the pointer-network (Vinyals et al., 2015) to copy tokens from the input to alleviate the out-of-vocabulary (OOV) issue. We adopt the calculation flows for copying mechanism from Yin et al. (2018b), readers can refer to that paper for further details.

B Marginal Distributions

To estimate the marginal distributions $p(\mathbf{x})$ and $q(\mathbf{y})$, we learn the LSTM language models over the NL utterances and MRs. MRs are linearized. Suppose given the NL $\mathbf{x} = \{x_i\}_{i=1}^{|\mathbf{x}|}$, the learning objective is:

$$p(\mathbf{x}) = \prod_{i=1}^{|\mathbf{x}|} p(x_i|x_{<i}) \tag{16}$$

where $x_{<i} = x_1 \cdots x_{i-1}$. At time t, we have the following probability to predict x_t:

$$p(x_t|x_{<t}) = f_{\text{softmax}}(\mathbf{W} \cdot \mathbf{h}_t + \mathbf{b}) \tag{17}$$

Here, $\mathbf{h}_t$ is estimated using the LSTM network:

$$\mathbf{h}_t = f_{\text{LSTM}}(x_t, \mathbf{h}_{t-1}) \tag{18}$$

The above marginal distribution estimation for NLs is also applied to linearized MRs.

C Experimental Setups

C.1 Marginal Distribution

We pre-train the language models on the full training set before joint learning and the language mdoels will be fixed in the following experiments. The embedding size is selected from $\{128, 256\}$ and the hidden size is tuned from $\{256, 512\}$, which are both evaluated on the validation set. We use SGD to

update the models. Early stopping is applied and the training will be stopped if the ppl value does not decrease for continuous 5 times.

C.2 Model Configuration

To conduct the joint learning using DIM and SEMIDIM, we have to firstly train the parser and generator separately referred as the method of SUPER.

To pre-train the parser and generator, we tune the embedding size from $\{125, 150, 256\}$ and hidden size from $\{256, 300, 512\}$. The batch size is selected from $\{10, 16\}$ varying over the datasets. Early stopping is applied and the patience time is set to 5. Initial learning rate is 0.001. Adam is adopted to optimize the models. The parser and generator will be trained until convergence.

After the pre-training, we conduct joint learning based on the pre-trained parser and generator. The learning rate will be slowed down to 0.00025. The beam size for sampling is tuned from $\{3, 5\}$.