Meta-Information Guided Meta-Learning for Few-Shot Relation Classification

Sanghyun Seo

Mar 23, 2021

Department of Computer Engineering at Dongguk University

Artificial Intelligence Laboratory

shseo@dongguk.edu

Papers

- **Domain Adaptive Dialog Generation via Meta Learning**
  - Qian, Kun, and Zhou Yu. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019.
- **Meta-Information Guided Meta-Learning for Few-Shot Relation Classification**
  - Dong, Bowen, et al. Proceedings of the 28th International Conference on Computational Linguistics. 2020.

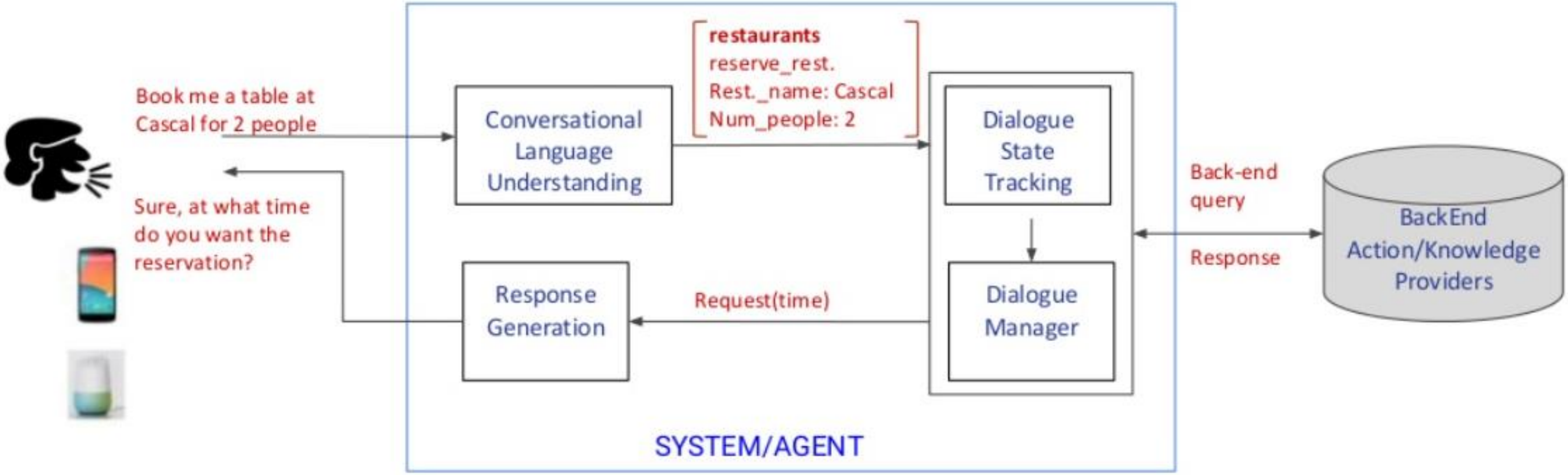
Background

- Dialogue System
  - Goal(task)-oriented
    - Personal assistant, helps users achieve a certain task
    - Goal: Task completion using combination of rules and learning
    - Examples:
      - End-to-end trainable task-oriented dialogue system (Wen et al., 2016)
      - End-to-end reinforcement learning dialogue system (Zhao and Eskenazi, 2016)
  - Chit-chat (open domain)
    - No specific goal, focus on natural responses
    - Goal: User engagement, naturalness, Using variants of seq2seq models
    - Examples:
      - A neural conversation model (Vinyals and Le, 2015)
      - Reinforcement learning for dialogue generation (Li et al., 2016)

Ref: https://www.slideshare.net/AIFrontiers/ai-frontiers-dilek-hakkanitur-conversational-machines-deep-learning-for-goaloriented-dialogue-systems

Background

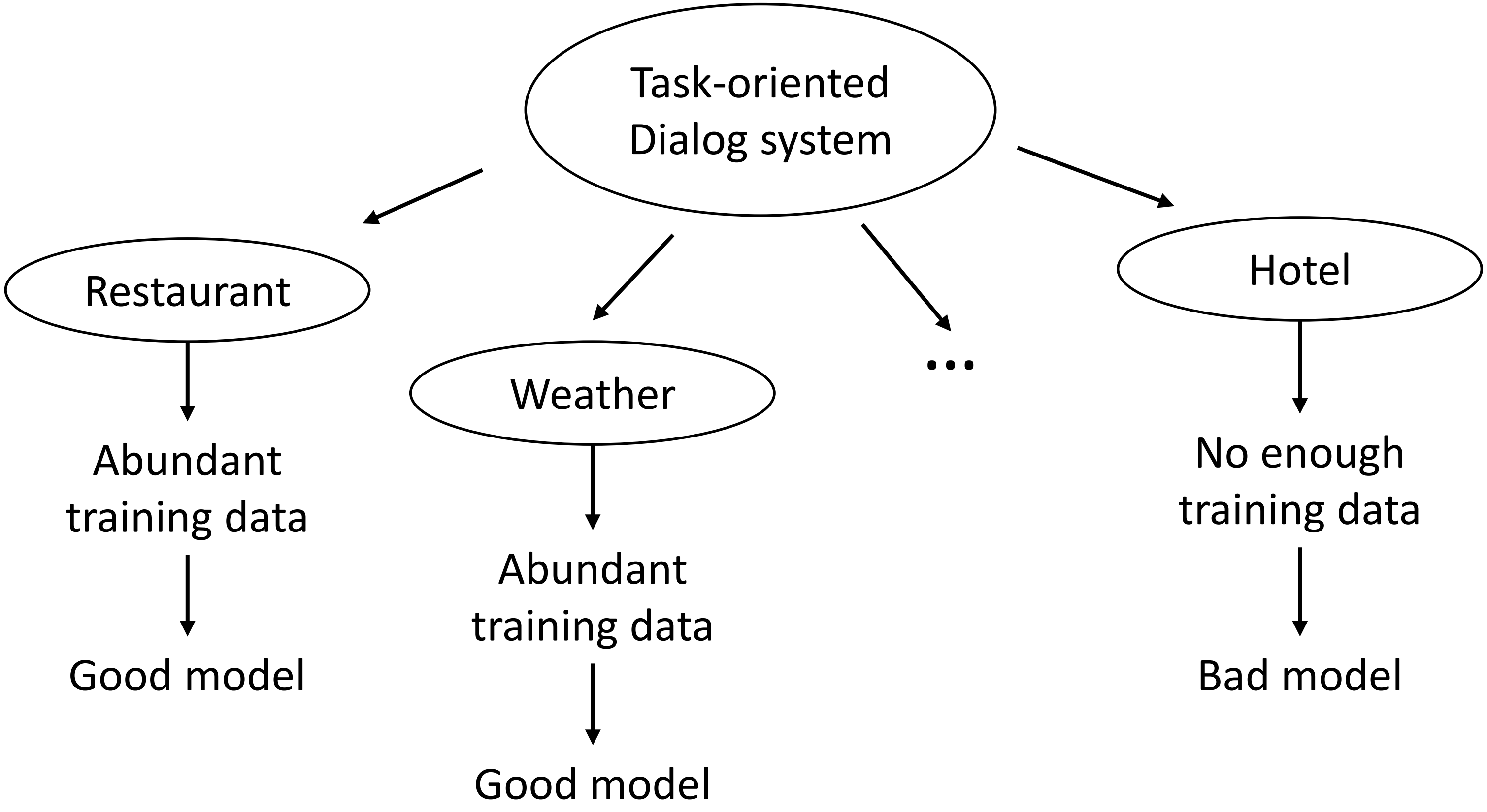
- Goal-Oriented Dialogue Systems
  - Conversational language understanding
  - Dialogue state tracking
  - Dialogue manager
  - Response generation



Ref: https://www.slideshare.net/AIFrontiers/ai-frontiers-dilek-hakkanitur-conversational-machines-deep-learning-for-goaloriented-dialogue-systems

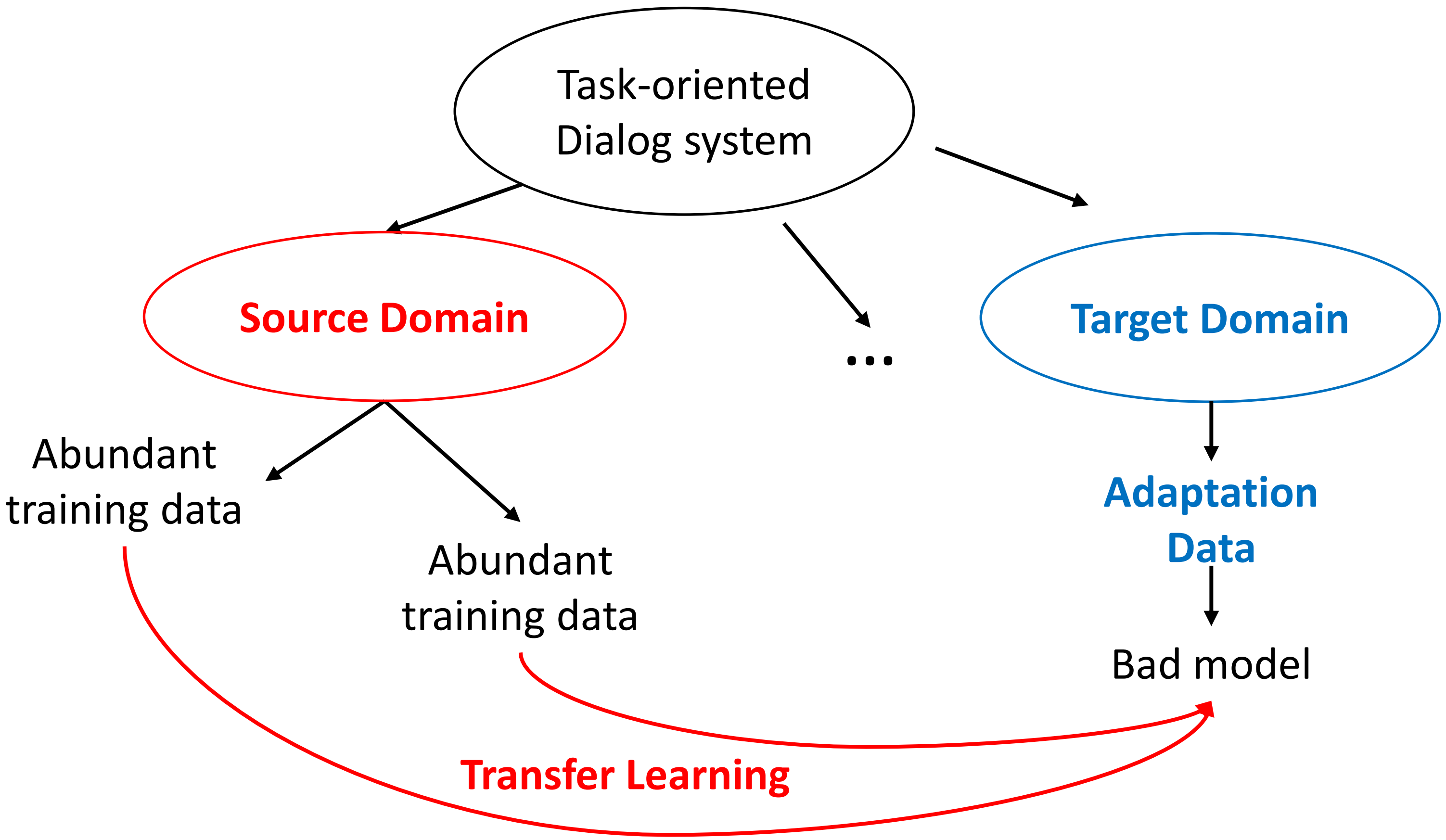
Background

- Low-resource learning



Background

- Low-resource learning



Background

- Traditional Transfer Learning
  - Pretrain on source domain

$$\min_{M} \sum_{K} Loss_k(M)$$

Best!

  - Fine-tune on target domain

$$M \leftarrow M - \alpha \nabla Loss_{k'}(M)$$

Not guaranteed!

- Model-agnostic meta learning
  - Pretrain on source domain

$$\min_{M} \sum_{K} Loss_k(M - \alpha \nabla Loss_k(M))$$

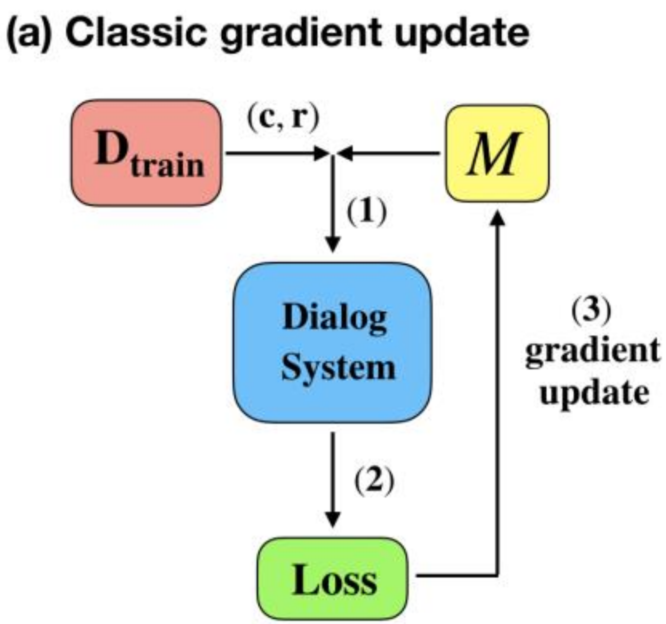
Maximum the efficiency of fin-tuning

  - Fine-tune on target domain

$$M \leftarrow M - \alpha \nabla Loss_{k'}(M)$$

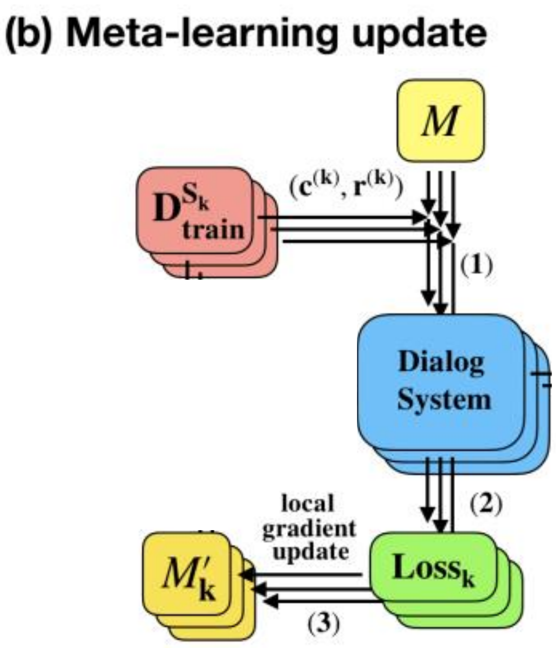
Methodology

- Classic gradient update for dialog system
  - 1) Apply model with sampled data (context, response) in dialog system
  - 2) compute loss
  - 3) update model with gradient descent
    $M \leftarrow M - \alpha \nabla Loss$

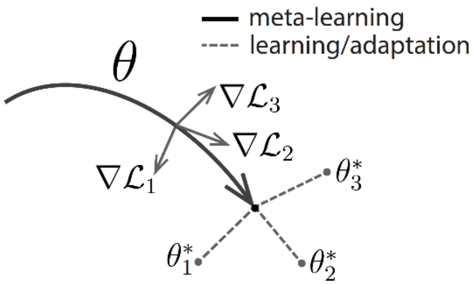


Methodology

- Meta-learning update
  - Initialized model M
  - for each domain S_k
    - 1) forward propagate with M
    - 2) Calculate the $Loss_k$
    - 3) $M'_k \leftarrow M - \alpha \nabla_M Loss(M, c^{(k)})$



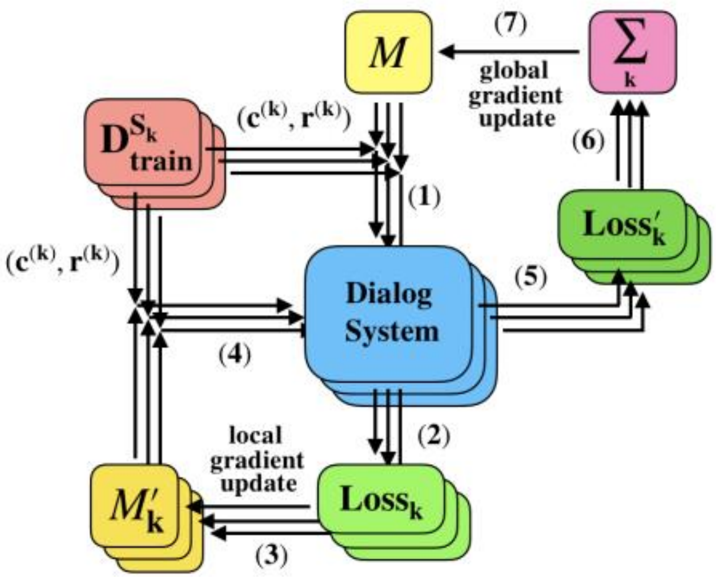
Methodology



- Meta-learning update
  - Initialized model M
  - for each domain S_k
    - 1) forward propagate with M
    - 2) Calculate the $Loss_k$
    - 3) $M_k' \leftarrow M - \alpha\nabla_M Loss(M, c^{(k)})$
    - 4) forward propagate with M_k'
    - 5) calculate the $Loss_k'$
  - 6) sum $Loss_k'$ over each domain
  - 7) update model M

$$M \leftarrow M - \beta\nabla_M \sum_K Loss_k(M_k', c^{(k)})$$

(b) Meta-learning update



Methodology

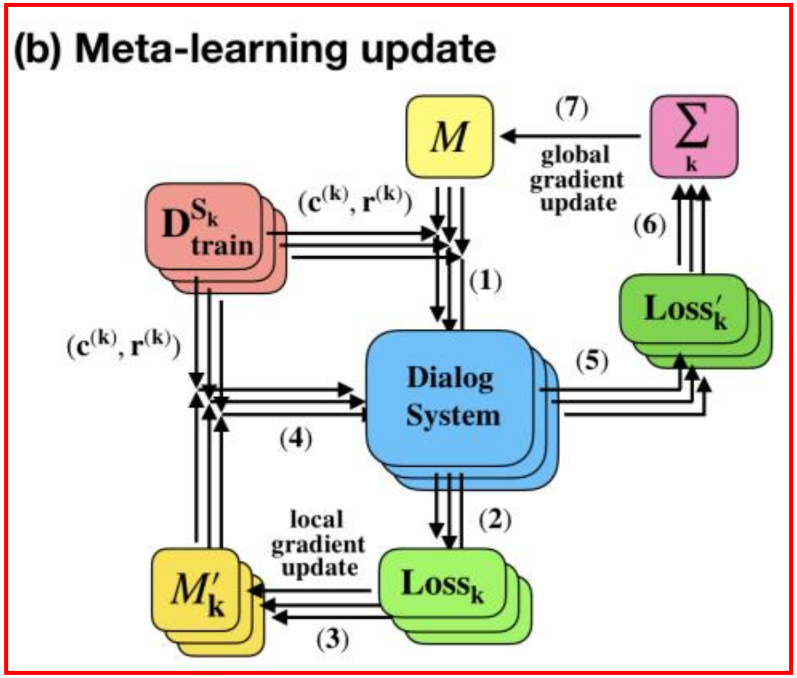
- DAML Algorithm

Algorithm 1 DAML

Input: dataset on source domain D^S_{train}; α; β
Output: optimal meta-learned model
  Randomly initialize model $\mathcal{M}$
  while *not done* **do**
    for $S_k \in Source\ Domain$ **do**
      Sample data $c^{(k)}$ from D^S_{train}
      $\mathcal{M}'_k = \mathcal{M} - \alpha \nabla_{\mathcal{M}} \mathcal{L}_{S_k}(\mathcal{M}, c^{(k)})$
      Evaluate $\mathcal{L}_{S_k}(\mathcal{M}'_k, c^{(k)})$
    end for
    $\mathcal{M} \leftarrow \mathcal{M} - \beta \nabla_{\mathcal{M}} \sum_{S_k} \mathcal{L}_{S_k}(\mathcal{M}'_k, c^{(k)})$
  end while

Function loss function $\mathcal{L}(\mathcal{M}, c)$
  return *cross-entropy*$(\mathcal{M}(c))$

Function $\mathcal{M}(c^{(k)} = \{B^{(k)}_{t-1}, R^{(k)}_{t-1}, U^{(k)}_t\})$
  $h = \text{Encoder}(B^{(k)}_{t-1}, R^{(k)}_{t-1}, U^{(k)}_t)$
  $B_t = \text{BspanDecoder}(h)$
  $R_t = \text{ResponseDecoder}(h, B^{(k)}_t, m^{(k)}_t)$
  return R_t



Methodology

- Dialog system model:
  - Sequicity

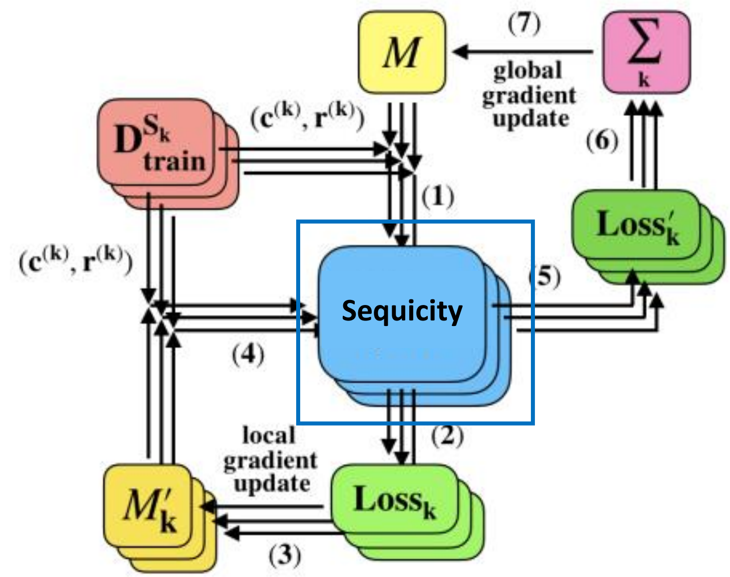
Algorithm 1 DAML

Input: dataset on source domain D^S_{train}; α; β
Output: optimal meta-learned model
  Randomly initialize model $\mathcal{M}$
  while *not done* **do**
    for $S_k \in Source\ Domain$ **do**
      Sample data $c^{(k)}$ from D^S_{train}
      $\mathcal{M}'_k = \mathcal{M} - \alpha \nabla_{\mathcal{M}} \mathcal{L}_{S_k}(\mathcal{M}, c^{(k)})$
      Evaluate $\mathcal{L}_{S_k}(\mathcal{M}'_k, c^{(k)})$
    end for
    $\mathcal{M} \leftarrow \mathcal{M} - \beta \nabla_{\mathcal{M}} \sum_{S_k} \mathcal{L}_{S_k}(\mathcal{M}'_k, c^{(k)})$
  end while

Function loss function $\mathcal{L}(\mathcal{M}, c)$
  return *cross-entropy*$(\mathcal{M}(c))$

Function $\mathcal{M}(c^{(k)} = \{B^{(k)}_{t-1}, R^{(k)}_{t-1}, U^{(k)}_t\})$
  $h = \text{Encoder}(B^{(k)}_{t-1}, R^{(k)}_{t-1}, U^{(k)}_t)$
  $B_t = \text{BspanDecoder}(h)$
  $R_t = \text{ResponseDecoder}(h, B^{(k)}_t, m^{(k)}_t)$
  return R_t

(b) Meta-learning update



Methodology

- Sequicity
  - copy-attention mechanism → Sequicity

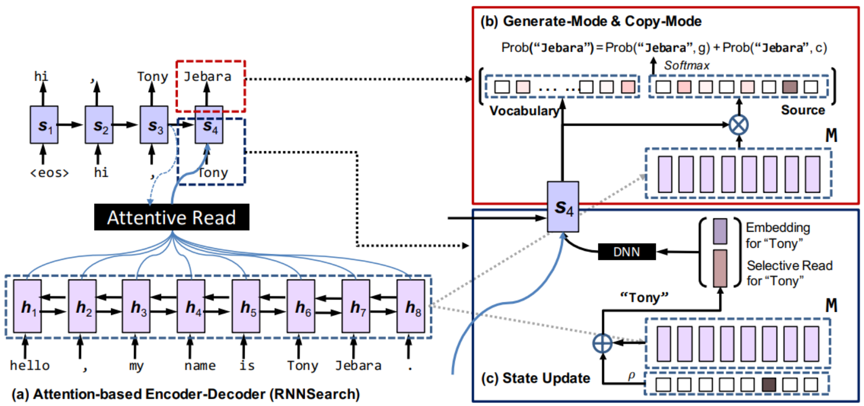


Figure 1: The overall diagram of COPYNET. For simplicity, we omit some links for prediction (see Sections 3.2 for more details).

I: Hello Jack, my name is Chandralekha.
R: Nice to meet you, Chandralekha.

I: This new guy doesn't perform exactly as we expected.
R: What do you mean by "doesn't perform exactly as we expected"?

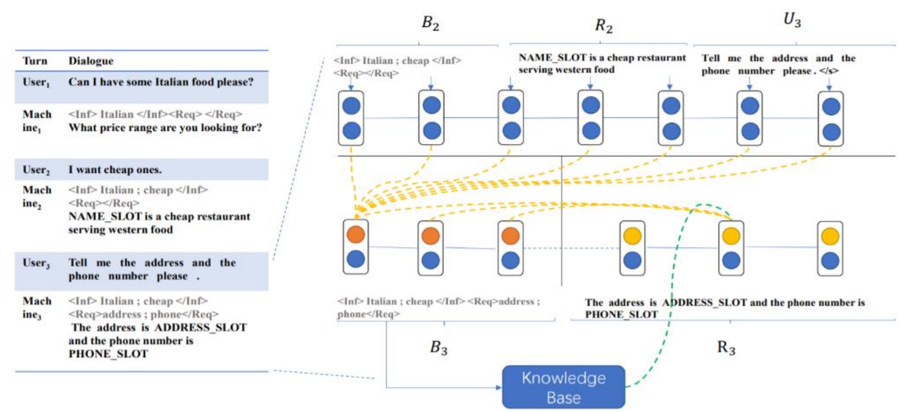


Figure 1: Sequicity overview. The left shows a sample dialogue; the right illustrates the Sequicity. B_t is employed only by the model, and not visible to users. During training, we substitute slot values with placeholders bearing the slot names for machine response. During testing, this is inverted: the placeholders are replaced by actual slot values, according to the item selected from the knowledge base.

Ref: Gu, Jiatao, et al. "Incorporating copying mechanism in sequence-to-sequence learning." *arXiv preprint arXiv:1603.06393* (2016).

Methodology

- Sequicity
  - two-step copy model

$$B_t = seq2seq(B_{t-1}, R_{t-1}, U_t)$$
$$R_t = seq2seq(B_{t-1}, R_{t-1}, U_t | B_t, m_t)$$

    B=belief span, R=response, U=utterance
    context $c = \{B_{t-1}, R_{t-1}, U_t\}$

  - structure of dialog system

$$h = Encoder(B_{t-1}, R_{t-1}, U_t)$$
$$B_t = BspanDecoder(h)$$
$$R_t = ResponseDecoder(h, B_t, m_t)$$

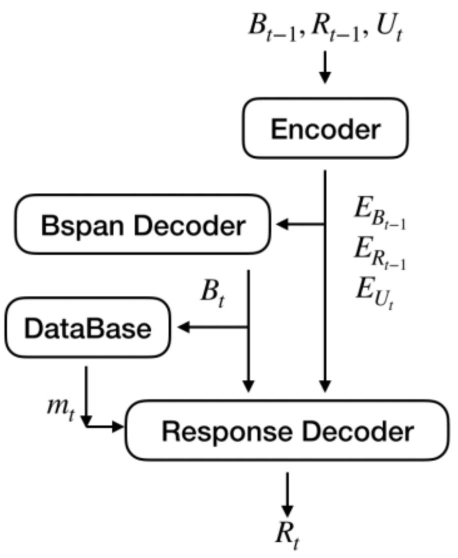


Figure 2: Structure of dialog system

Methodology

- Sequicity
  - structure of dialog system
    - m_t = **'no match'**
      → the system would restart the conversation
    - m_t = **'exact match'**
      → the system successfully retrieves
      the requested information
      and completes the tasks
    - m_t = **'multiple match'**
      → the system will then output
      a question to elicit more information

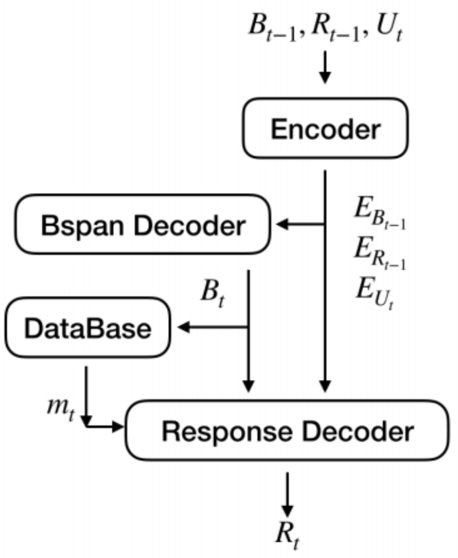


Figure 2: Structure of dialog system

Experiments

- Dataset: Simdial (Zhao and Eskenazi, 2018)
  - Source domains
    - 900 training, 100 validation dialogs for each domain
    - Domains: Restaurant, Bus, Weather
  - Target domains
    - 9 adaptation, 500 testing dialogs for each domain
    - Restaurant **(in-domain)**
    - Restaurant-slot **(unseen slot)**: new slot value
    - Restaurant-style **(unseen NLG)**: same slot values but different NLG templates
    - Movie **(new-domain)**: completely new domains
  - Metric
    - BLEU score: quality of generated response sentence
    - Entity F1 score: completeness of tasks
    - number of epochs: adaptation efficiency

Experiments

- Dataset: Simdial (Zhao and Eskenazi, 2018)
  - Example

turn	speaker	utterances	inform slots	request slots
1	**user** **sys**	What's up? hmm I am looking for a restaurant. Which place?		
2	**user** **sys**	I uhm yeah I don't care. Oh sorry, Philadelphia. I believe you said Philadelphia.	loc,Philadelphia	
3	**user** **sys**	I have more requests. What kind of parking does it have? The restaurant has no parking. Anything else?	loc,Philadelphia; food,Indian	parking
4	**user** **sys**	I have more requests. Is hmm ... it closed? No, It is open right now. What else can I do?	loc,Philadelphia; food,Indian	opening
5	**user** **sys**	New request. I'm interested in food uhm at Seattle. Do you mean Indian?	loc,Seattle; food,Indian	
6	**user** **sys**	Uh-huh. Restaurant 56 is a good choice. What else can I do?	loc,Seattle; food,Indian	
7	**user** **sys**	Not done yet. What's the average price? The restaurant serves moderate food.	loc,Seattle; food,Indian	price
8	**user** **sys**	I have all I need. See you. See you next time.	loc,Seattle; food,Indian	

Table 3: An example dialog generated from SimDial

Experiments

- Experiments results
  - few-shot dialogue generation

In Domain	ZSDG	Transfer	DAML	Transfer-oneshot	DAML-oneshot
BLEU	70.1	51.8	51.8	51.1	53.7
Entity F1	79.9	88.5	**91.4**	87.6	91.2
Epoch	-	2.7	1.4	2.2	1.0
Unseen Slot	ZSDG	Transfer	DAML	Transfer-oneshot	DAML-oneshot
BLEU	68.5	43.3 (46.3)	41.7 (46.3)	40.8 (43.9)	40.0 (41.8)
Entity F1	74.6	78.7 (78.5)	75 (**79.2**)	70.1 (67.7)	72.0 (73.0)
Epoch	-	2.6 (2.4)	4.8 (3.4)	3.2 (2.6)	5.0 (3.0)
Unseen NLG	ZSDG	Transfer	DAML	Transfer-oneshot	DAML-oneshot
BLEU	70.1	30.6 (32.4)	21.5 (26.0)	20.0 (21.5)	19.1 (19.1)
Entity F1	72.9	82.2 (85.0)	77.5 (82.4)	82.8 (86.2)	69.0 (**86.4**)
Epoch	-	3.2 (3.0)	3.2 (2.1)	12.3 (20.3)	4.7 (5.7)
New Domain	ZSDG	Transfer	DAML	Transfer-oneshot	DAML-oneshot
BLEU	54.6	30.1	32.7	21.5	22.4
Entity F1	52.6	64.0	**66.2**	55.9	59.5
Epoch	-	5.6	4.5	14.2	5.8

Table 1: DAML outperforms both ZSDG and transfer learning when given similar target domain data. Even the one-shot DAML method achieves better results than ZSDG. Values in parenthesis are the results of the model with an extra step of fine-tuning on the restaurant domain in training. “In Domain” uses all three source domains (*restaurant*, *weather* and *bus*), while “New Domain” refers to the *movie* domain. “Unseen Slot” and “Unseen NLG” correspond to *restaurant-slot* and *restaurant-style* separately.

Experiments

- Experiments results
  - leave-one-out approach
  - Impact of using different amount of target domain data on system performance

movie	Transfer	DAML
Entity F1	64.0	**66.2**
BLEU	30.1	**32.7**
restaurant	Transfer	DAML
Entity F1	80.7	**82.1**
BLEU	46.1	**47.9**
bus	Transfer	DAML
Entity F1	60.0	**61.9**
BLEU	32.0	**35.9**
weather	Transfer	DAML
Entity F1	79.1	**80.4**
BLEU	38.9	**43.3**

Table 2: Performance on different dialog domains

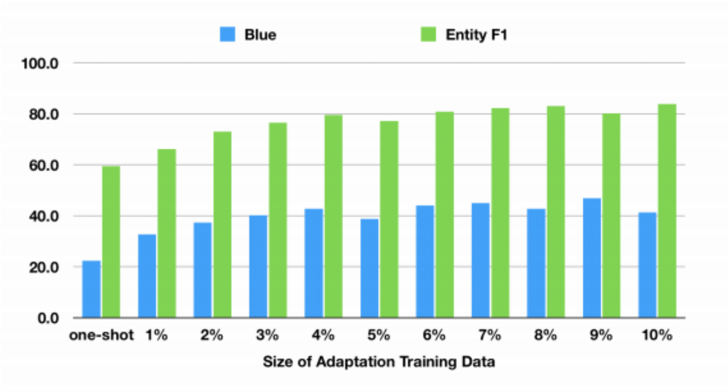
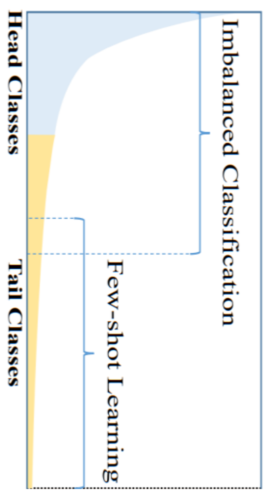
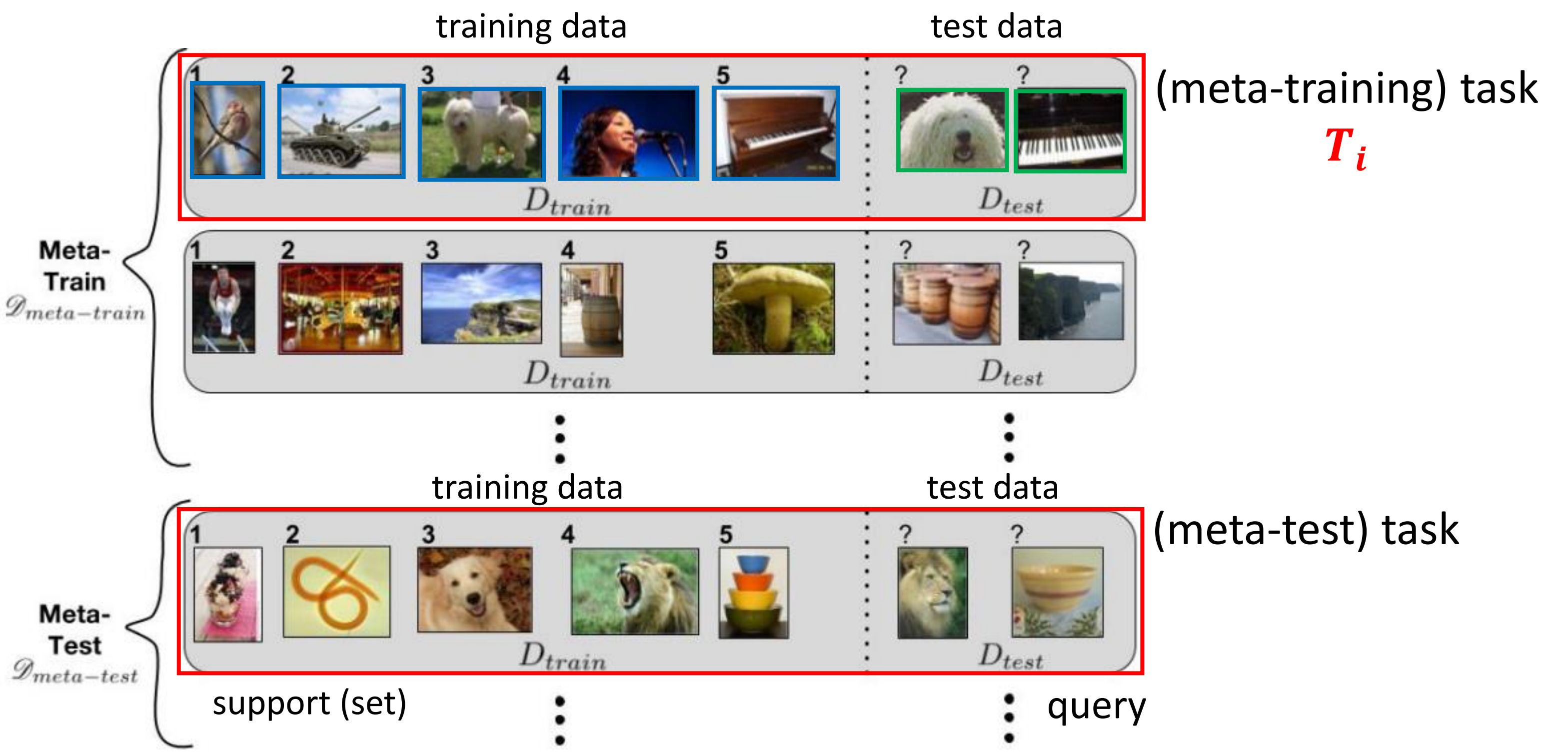


Figure 3: The system performance improves when the size of the target data increases. Even the one-shot learning setting achieves decent performance.

Recap: FSL Setting

- Meta-learning setup
  - $(D_{train}$ / $D_{test})$ / $(D_{train}$ / $D_{test})$ ← Meta-Train / Meta-Test
  - $D_i^{tr} = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}, D_i^{ts} = \{(x_1^i, y_1^i), \dots, (x_l^i, y_l^i)\}$
  - Task(episode) $T_i = \{D_i^{tr}, D_i^{ts}\}$





Ravi, Sachin, and Hugo Larochelle. "Optimization as a model for few-shot learning." (2016).

Task in few-shot NLP

- Domain as task
  - **ARSC:** multi-domain sentiment classification
    - 23 domains, 3 binary classification tasks
      → total 69 tasks (12 tasks, 4 domains are target tasks)
  - **CNICN150:** multi-domain intent classification
    - 10 domains, 15 intents (total 150 intents)
      → 22,500 labeled example, 1200 out-of-scope instances)

- Class as task
  - **FewRel:** few-shot relation classification
    - 100 relations (tr:64/dev:16/te:20), same domain(Wikipedia corpus and Wikidata knowledge bases)
      → FewRel 2.0 added a new domain of test set and 'none-of-above' relation
  - **SNIPS:** few-shot intent classification
    - 7 intents (tr:5/te:2)

Yin, Wenpeng. "Meta-learning for few-shot natural language processing: A survey." arXiv preprint arXiv:2007.09604 (2020).

Background

- Challenges of meta learning
  - Using only support set to classify query set
    - Most meta-learning methods learn how to learn (i.e., how to initialize and adapt) **solely relying on instance statistics**, which inevitably suffer from data sparsity and noise in low-resource scenarios, especially in text domain
  - Lack of interpretability
    - The approach of learning to learn, like the learning process itself, is a black-box and thus **lacks interpretability**
  - Weakness of zero-shot learning
    - Most conventional meta-learning methods are designed for few-shot classification, and **cannot well handle zero-shot scenarios**, where no support instances are available

Methodology

- MIML (Meta Information guided Meta Learning)
  1) Instance encoder
  2) Meta-information guided fast initialization
  3) Meta-information guided fast adaptation
  4) Meta-optimization

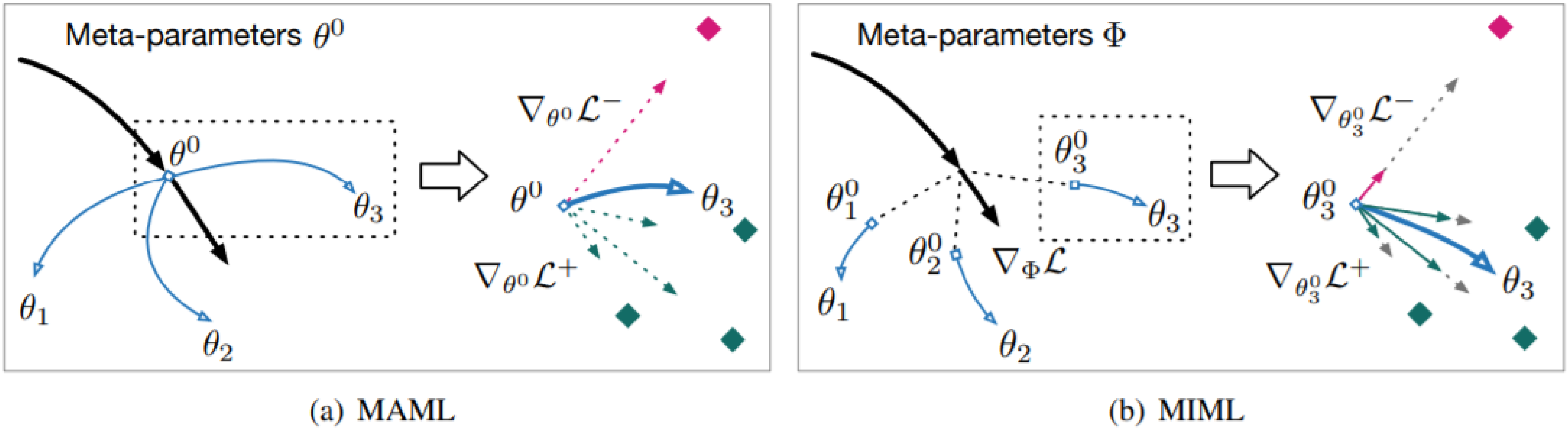


Figure 1: Diagram of meta-learning models. (a) MAML learns a class-agnostic representation θ^0 that can fast adapt to new classes. (b) MIML learns meta-parameters Φ to fast initialize class-aware parameter θ_i^0, and to quickly adapt to new classes using informative instances, where both phases are guided by meta-information. Informative instances and noisy instances are marked accordingly.

Methodology

- Instance encoder
  - BERT model to encode the instance into contextualized representations
$$\mathrm{x}_j = g\left(x_j, h, t; \phi_e\right)$$
  - x_j is the sentence, h and t are head and tail entities respectively. $g(\cdot)$ is the encoder, ϕ_e is the parameters of the encoder, and $\mathrm{x}_j \in \mathbb{R}^{d_s}$ is the instance representation

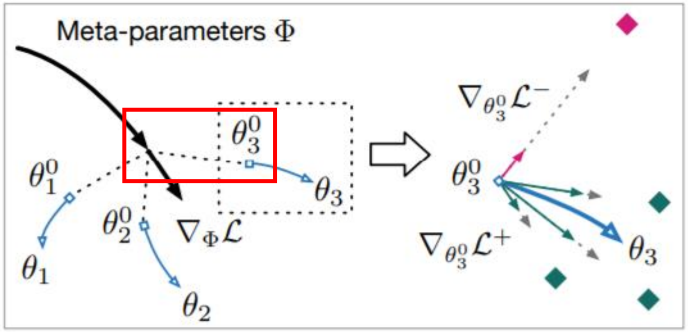
Methodology

- Meta-information guided fast initialization
  - Instead of using a static class-agnostic initialization point for all classes as in MAML, MIML uses meta-information to estimate **dynamic class-aware initialization parameters** for each class
  - This alleviates the reliance on support instances to reach optimal adapted parameters

Algorithm 1 Meta-Information Guided Meta-Learning

Require: $p(\mathcal{C})$: distribution over classes
Require: β: meta learning rate

randomly initialize:
    $\Phi = \{\phi_e, \phi_n, \phi_a\}$: meta-parameters
while not done **do**
    Sample batch of classes $\mathcal{C}_i \sim p(\mathcal{C})$
    Sample support instance set $\mathcal{S}$ and query instance set $\mathcal{Q}$
    for all $\mathcal{C}_i$ **do**
        Fast initialize parameters of $\mathcal{C}_i$: $\theta_i^0 = \Psi(\mathbf{c}_i; \phi_n)$
    for $t = 1, \ldots, T$ **do**
        Compute gradients and learning rates for fast adaptation using support instance set $\mathcal{S}$
        Compute adapted parameters with gradient descent: $\theta^{t+1} = \theta^t - \sum_{i,j} \alpha_{i,j} \nabla_{\theta^t} \mathcal{L}(f_{\theta^t,\{\phi_e,\phi_n\}}, x_j, y_j)$
    Meta-optimize using query instance set $\mathcal{Q}$: $\Phi = \Phi - \beta \nabla_\Phi \mathcal{L}(f_{\theta^T,\Phi}, x_j, y_j)$



(b) MIML

Methodology

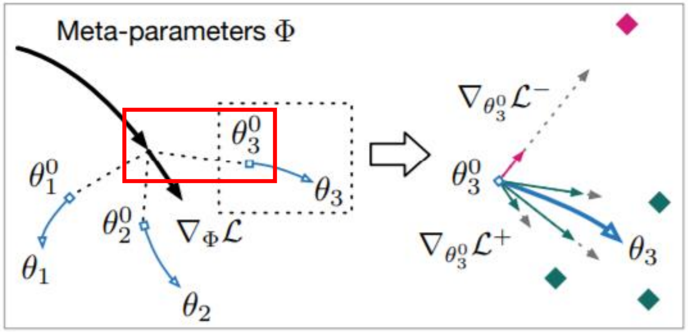
- Meta-information guided fast initialization
  - Given the name of a class C_i, the meta-information representation $c_i \in \mathbb{R}^{d_w}$ is obtained by **the average of the word embeddings of the name**

$$\theta_i^0 = \Psi(c_i; \phi_n)$$

  - where $\theta_i^0 \in \mathbb{R}^{d_s}$ is the class-aware initialization parameters for class C_i, $\Psi(c_i; \phi_n)$ is the **meta-initializer**, ϕ_n is the corresponding meta-parameters

Algorithm 1 Meta-Information Guided Meta-Learning

Require: $p(\mathcal{C})$: distribution over classes
Require: β: meta learning rate
randomly initialize:
  $\Phi = \{\phi_e, \phi_n, \phi_a\}$: meta-parameters
while not done **do**
  Sample batch of classes $\mathcal{C}_i \sim p(\mathcal{C})$
  Sample support instance set $\mathcal{S}$ and query instance set $\mathcal{Q}$
  for all $\mathcal{C}_i$ **do**
    Fast initialize parameters of $\mathcal{C}_i$: $\theta_i^0 = \Psi(\mathbf{c}_i; \phi_n)$
  for $t = 1, \ldots, T$ **do**
    Compute gradients and learning rates for fast adaptation using support instance set $\mathcal{S}$
    Compute adapted parameters with gradient descent:
    $\theta^{t+1} = \theta^t - \sum_{i,j} \alpha_{i,j} \nabla_{\theta^t} \mathcal{L}(f_{\theta^t,\{\phi_e,\phi_n\}}, x_j, y_j)$
  Meta-optimize using query instance set $\mathcal{Q}$:
  $\Phi = \Phi - \beta \nabla_\Phi \mathcal{L}(f_{\theta^T,\Phi}, x_j, y_j)$



(b) MIML

Methodology

- Meta-information guided fast initialization
  - $\Psi(\cdot)$ is implemented via a fully connected layer
  - It usually is a rough in an early stage, but flexible estimation of a new concept based on its high-level semantics is possible

$$s_{i,j} = {\theta_i^0}^T x_j$$

  - where $s_{i,j}$ is the score of x_j being an instance of C_i. The probability $p(y = C_i|x_j)$ is obtained by normalizing the score $s_{i,j}$ with a softmax layer over all classes $\{C_1, C_2, \ldots, C_N\}$
  - The model after fast initialization can be denoted as $f_{\theta_0,\{\phi_e,\phi_n\}}$, where $\theta^0 = \{\theta_1^0, \theta_2^0, \ldots, \theta_N^0\}$ denotes initialized parameters

Methodology

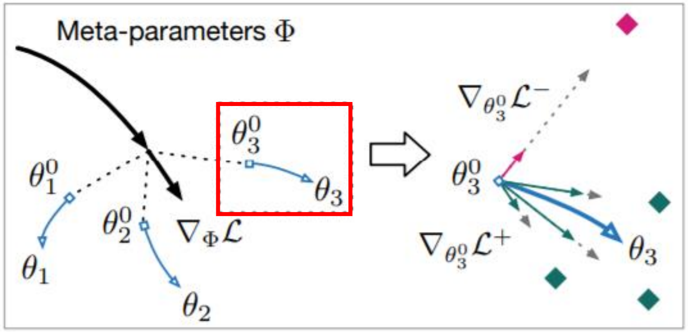
- Meta-information guided fast adaptation
  - The initialized parameters θ^0 are adapted via gradient descent steps according to the classification performance of instances on the support set S
  - The adaptation iterates dynamically for T steps

$$\theta^{t+1} = \theta^t - \sum_{i,j} \alpha_{i,j} \nabla_{\theta^t} \mathcal{L}(f_{\theta_{0,\{\phi_e,\phi_n\}}}, x_j, y_j)$$

  - $\mathcal{L}(\cdot)$ denotes cross-entropy loss of a support instance

Algorithm 1 Meta-Information Guided Meta-Learning

Require: $p(\mathcal{C})$: distribution over classes
Require: β: meta learning rate
randomly initialize:
  $\Phi = \{\phi_e, \phi_n, \phi_a\}$: meta-parameters
while not done **do**
  Sample batch of classes $\mathcal{C}_i \sim p(\mathcal{C})$
  Sample support instance set $\mathcal{S}$ and query instance set $\mathcal{Q}$
  for all $\mathcal{C}_i$ **do**
    Fast initialize parameters of $\mathcal{C}_i$: $\theta_i^0 = \Psi(\mathbf{c}_i; \phi_n)$
  for $t = 1, \ldots, T$ **do**
    Compute gradients and learning rates for fast adaptation using support instance set $\mathcal{S}$
    Compute adapted parameters with gradient descent: $\theta^{t+1} = \theta^t - \sum_{i,j} \alpha_{i,j} \nabla_{\theta^t} \mathcal{L}(f_{\theta^t,\{\phi_e,\phi_n\}}, x_j, y_j)$
  Meta-optimize using query instance set $\mathcal{Q}$:
  $\Phi = \Phi - \beta \nabla_\Phi \mathcal{L}(f_{\theta^T,\Phi}, x_j, y_j)$



(b) MIML

Methodology

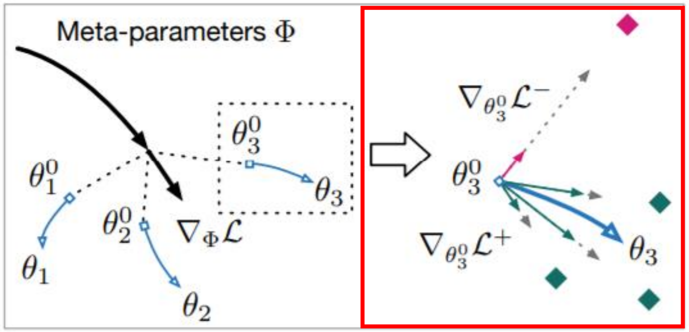
- Meta-information guided fast adaptation
  - To select informative instances for fast adaptation in MIML, instead of using a static learning rate for all instances, the learning rate of each instance is dynamically determined by a selective attention mechanism as follows:

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_j \exp(e_{i,j})}$$

  - where $e_{i,j}$ is the score of instance x_j for class C_i.

Algorithm 1 Meta-Information Guided Meta-Learning

Require: $p(\mathcal{C})$: distribution over classes
Require: β: meta learning rate
randomly initialize:
 $\Phi = \{\phi_e, \phi_n, \phi_a\}$: meta-parameters
while not done **do**
 Sample batch of classes $\mathcal{C}_i \sim p(\mathcal{C})$
 Sample support instance set $\mathcal{S}$ and query instance set $\mathcal{Q}$
 for all $\mathcal{C}_i$ **do**
  Fast initialize parameters of $\mathcal{C}_i$: $\theta_i^0 = \Psi(\mathbf{c}_i; \phi_n)$
 for $t = 1, \ldots, T$ **do**
  Compute gradients and learning rates for fast adaptation using support instance set $\mathcal{S}$
  Compute adapted parameters with gradient descent:
  $\theta^{t+1} = \theta^t - \sum_{i,j} \alpha_{i,j} \nabla_{\theta^t} \mathcal{L}(f_{\theta^t, \{\phi_e, \phi_n\}}, x_j, y_j)$
 Meta-optimize using query instance set $\mathcal{Q}$:
 $\Phi = \Phi - \beta \nabla_\Phi \mathcal{L}(f_{\theta^T, \Phi}, x_j, y_j)$



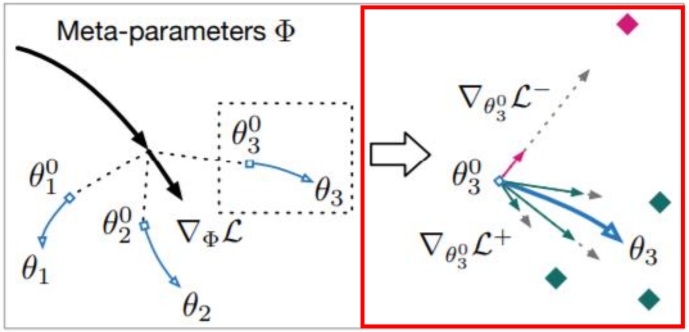
(b) MIML

Methodology

- Meta-information guided fast adaptation
  - The score is obtained by:
    $$e_{i,j} = {q_i}^T x_j$$
  - Where $q_i \in \mathbb{R}^{d_s}$ is the query vector for class C_i
  - Estimating the query vector from meta-information via a meta-querier module as follows:
    $$q_i = \Psi(c_i; \phi_a)$$

Algorithm 1 Meta-Information Guided Meta-Learning

Require: $p(\mathcal{C})$: distribution over classes
Require: β: meta learning rate
randomly initialize:
    $\Phi = \{\phi_e, \phi_n, \phi_a\}$: meta-parameters
while not done **do**
    Sample batch of classes $\mathcal{C}_i \sim p(\mathcal{C})$
    Sample support instance set $\mathcal{S}$ and query instance set $\mathcal{Q}$
    for all $\mathcal{C}_i$ **do**
        Fast initialize parameters of $\mathcal{C}_i$: $\theta_i^0 = \Psi(\mathbf{c}_i; \phi_n)$
    for $t = 1, \dots, T$ **do**
        Compute gradients and learning rates for fast adaptation using support instance set $\mathcal{S}$
        Compute adapted parameters with gradient descent:
        $\theta^{t+1} = \theta^t - \sum_{i,j} \alpha_{i,j} \nabla_{\theta^t} \mathcal{L}(f_{\theta^t, \{\phi_e, \phi_n\}}, x_j, y_j)$
    Meta-optimize using query instance set $\mathcal{Q}$:
    $\Phi = \Phi - \beta \nabla_\Phi \mathcal{L}(f_{\theta^T, \Phi}, x_j, y_j)$



(b) MIML

Methodology

- Meta-information guided fast adaptation
  - The score is obtained by:
    $$e_{i,j} = {q_i}^T x_j$$
  - where $q_i \in \mathbb{R}^{d_s}$ is the query vector for class C_i
  - The estimated query vector from meta-information via a **meta-querier module** as follows:
    $$q_i = \Psi(c_i; \phi_a)$$
- Overfitting Problem
  - **L2 normalization**
  - Virtual adversarial training

Methodology

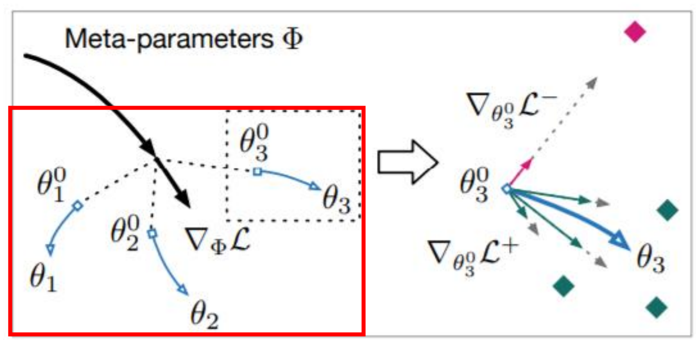
- Meta-optimization
  - After fast adaptation on support instances, the meta-parameters $\Phi = \{\phi_e, \phi_n, \phi_a\}$ are optimized according to the performance of the adapted model on the query set Q as follows:

$$\Phi = \Phi - \beta \nabla_{\Phi} \mathcal{L}(f_{\theta^T,\Phi}, x_j, y_j)$$

  - where β is the learning rate for meta-parameters

Algorithm 1 Meta-Information Guided Meta-Learning

Require: $p(\mathcal{C})$: distribution over classes
Require: β: meta learning rate
randomly initialize:
  $\Phi = \{\phi_e, \phi_n, \phi_a\}$: meta-parameters
while not done **do**
  Sample batch of classes $\mathcal{C}_i \sim p(\mathcal{C})$
  Sample support instance set $\mathcal{S}$ and query instance set $\mathcal{Q}$
  for all $\mathcal{C}_i$ **do**
    Fast initialize parameters of $\mathcal{C}_i$: $\theta_i^0 = \Psi(\mathbf{c}_i; \phi_n)$
  for $t = 1, \ldots, T$ **do**
    Compute gradients and learning rates for fast adaptation using support instance set $\mathcal{S}$
    Compute adapted parameters with gradient descent:
    $\theta^{t+1} = \theta^t - \sum_{i,j} \alpha_{i,j} \nabla_{\theta^t} \mathcal{L}(f_{\theta^t, \{\phi_e, \phi_n\}}, x_j, y_j)$
  Meta-optimize using query instance set $\mathcal{Q}$:
  $\Phi = \Phi - \beta \nabla_{\Phi} \mathcal{L}(f_{\theta^T,\Phi}, x_j, y_j)$



(b) MIML

Methodology

- Implementation Details
  - Model: $BERT_{base}$ with GloVe 50d word embeddings
  - class distribution $p(C)$: uniform distribution
  - # of adaptation step: 150
  - optimizer: Adam

- Dataset & Evaluation Protocol
  - Dataset: FewRel(70,000 labeled sentences in 100 relations)
  - Evaluation: 5-way 1-shot, 5-way 5-shot, 10-way 1-shot, 10-way 5-shot.
  - Baseline: MetaNets, GNN, SNAIL, ProtoNets, MLMAN, BERT-PAIR, ProtoNets(with BERT encoder), MAML(with BERT encoder)

Experiments

- Main results
  - Meta-information guided fast initialization in MIML can produce more flexible class-aware initialization, which alleviates heavy reliance on support instances

Encoder	Model	5-way-1-shot	5-way-5-shot	10-way-1-shot	10-way-5-shot
CNN	Meta Network*	64.46 ± 0.54	80.57 ± 0.48	53.96 ± 0.56	69.23 ± 0.52
	GNN*	66.23 ± 0.75	81.28 ± 0.62	46.27 ± 0.80	64.02 ± 0.77
	SNAIL*	67.29 ± 0.26	79.40 ± 0.22	53.28 ± 0.27	68.33 ± 0.25
	Proto Network*	74.52 ± 0.07	88.40 ± 0.06	62.38 ± 0.06	80.45 ± 0.08
	MLMAN*	82.98 ± 0.20	92.66 ± 0.09	73.59 ± 0.26	87.29 ± 0.15
BERT	BERT-PAIR ♠	88.32 ± 0.64	93.22 ± 0.13	80.63 ± 0.17	87.02 ± 0.12
	MAML	87.45 ± 0.11	94.39 ± 0.13	78.91 ± 0.14	89.14 ± 0.23
	Proto Network	86.50 ± 0.14	95.01 ± 0.15	82.86 ± 0.15	91.30 ± 0.11
	MIML	$\mathbf{92.55 \pm 0.12}$	$\mathbf{96.03 \pm 0.17}$	$\mathbf{87.47 \pm 0.21}$	$\mathbf{93.22 \pm 0.22}$
-	Human*	92.22	-	85.88	-

Table 1: Main results. Accuracies (%) on few-shot relation classification on FewRel test set. Results with * and ♠ are from FewRel leaderboard and Gao et al. (2019b) respectively.

Experiments

- Robustness to Noisy Instances
  - Randomly corrupt 0%, 10%, 20%, 30% support instances, by replacing them with noisy instances randomly sampled from different relations in FewRel

Model	Noise Rate	5-way-5-shot	10-way-5-shot	Noise Rate	5-way-5-shot	10-way-5-shot
MAML	0%	92.59 ± 0.08	85.79 ± 0.15	10%	90.81 ± 0.12	83.31 ± 0.13
Proto Network		92.62 ± 0.11	87.12 ± 0.12		91.54 ± 0.08	85.40 ± 0.18
Proto HATT		93.43 ± 0.09	89.37 ± 0.17		92.40 ± 0.13	88.19 ± 0.22
MIML		$\mathbf{95.60 \pm 0.09}$	$\mathbf{91.60 \pm 0.21}$		$\mathbf{94.82 \pm 0.08}$	$\mathbf{89.55 \pm 0.25}$
MAML	20%	88.40 ± 0.10	80.77 ± 0.13	30%	86.18 ± 0.20	78.30 ± 0.11
Proto Network		91.04 ± 0.08	83.18 ± 0.17		87.84 ± 0.12	80.28 ± 0.19
Proto HATT		91.27 ± 0.15	85.94 ± 0.29		89.62 ± 0.19	83.14 ± 0.24
MIML		$\mathbf{93.19 \pm 0.10}$	$\mathbf{87.70 \pm 0.23}$		$\mathbf{92.04 \pm 0.18}$	$\mathbf{86.19 \pm 0.27}$

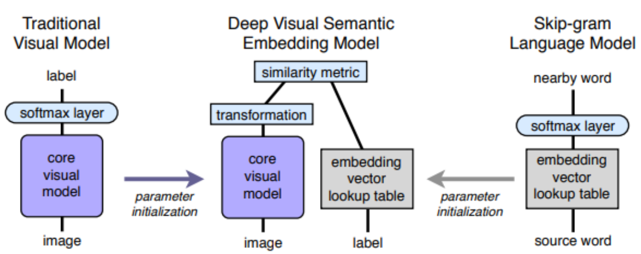
Table 2: Accuracies (%) on few-shot relation classification with noise on FewRel development set.

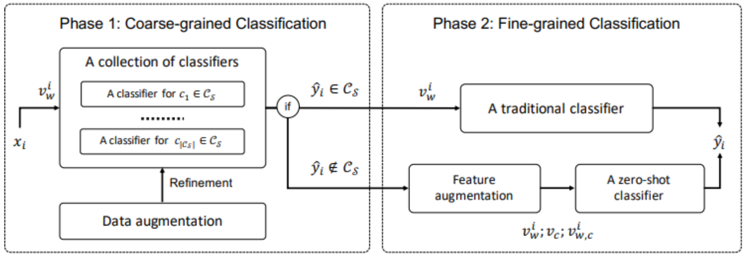
Experiments

- Zero-Shot Classification
  - Remove the support instances in evaluation phase in 5-way and 10-way setting, and ask the model to classify query instances with class-aware initialization parameters
    - DeViSE model with BERT encoder
    - SK4 with rich semantic knowledge of classes, including word embeddings, class descriptions, class hierarchy, and commonsense knowledge graphs

Setting	Random	DeViSE	SK4	MIML
5-way-0-shot	20.00	55.90 ± 0.09	$\mathbf{79.68 \pm 0.12}$	79.54 ± 0.06
10-way-0-shot	10.00	42.29 ± 0.08	$\mathbf{66.17 \pm 0.11}$	61.14 ± 0.10

Table 3: Experimental results of zero-shot classification on FewRel development set.





Experiments

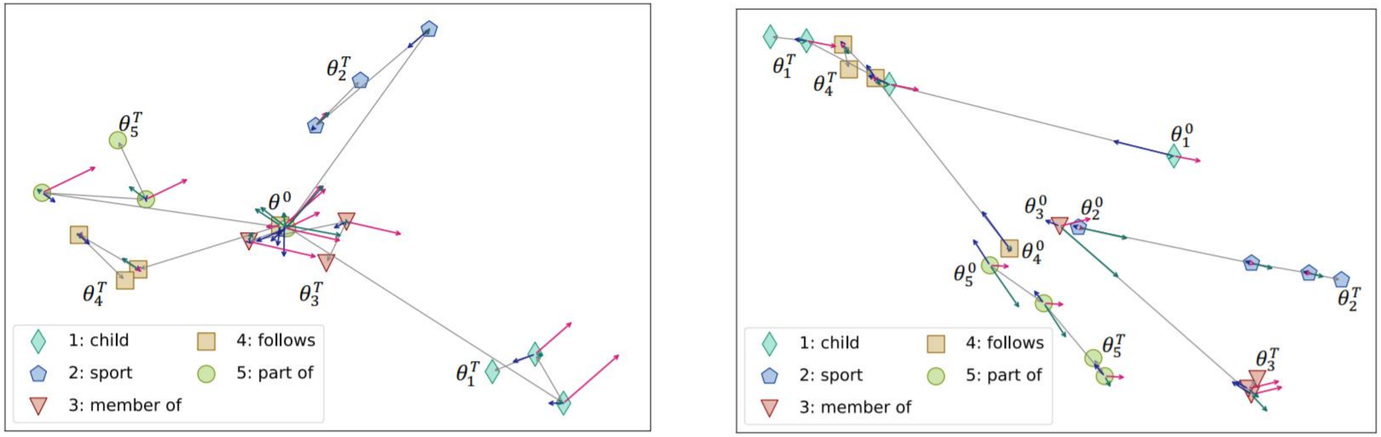
- Ablation Study
  - Ablation study in 10-way5-shot setting, by removing each component, including meta-information guided fast initialization (MI) and adaptation (MA), class-aware parameter normalization (NM) and virtual adversarial training (VAT)

Model	MAML	MIML	MIML w/o MI	MIML w/o MA	MIML w/o NM	MIML w/o VAT
Accuracy	85.79 ± 0.15	$\mathbf{91.60 \pm 0.21}$	86.43 ± 0.17	89.59 ± 0.19	84.17 ± 0.13	89.43 ± 0.09

Table 4: Ablation results in 10-way-5-shot setting on FewRel development set. MI/MA: meta-information guided fast initialization/adaptation, NM: Normalization, VAT: virtual adversarial training.

Experiments

- Visualization
  - Visualizing the workflow of MIML in the presence of 20% noise in 5-way-5-shot setting and comparing it with MAML
  - The initialization representations and adaptation steps are visualized by applying principal component analysis



(a) MAML (b) MIML

Figure 2: Visualization of initialization and adaptation process of meta-learning models, in 5-way-5-shot setting with 20% noise. At each iteration, the adaptation gradients for a class parameter θ_i come from three parts: informative instances from class C_i (marked in **green** arrows), noisy instance for class C_i (marked in **red** arrows), and instances for other classes (marked in **blue** arrows).[1] Best viewed in color.

Future Works

- Meta information
  - Exploring more meta-information for meta-learning, such as class descriptions and knowledge graphs
- Enhanced Encoder
  - Developing more sophisticated models to capture the fine-grained interactions between the high-level meta information and concrete instances, to better guide meta-learning for few-shot classification problem
- Hybrid approach for meta learning
  - Integrating optimization-based approaches and metric-based approaches to make a better performance, to do few-shot classification and zero-shot classification simultaneously

Q&A
Thank you!