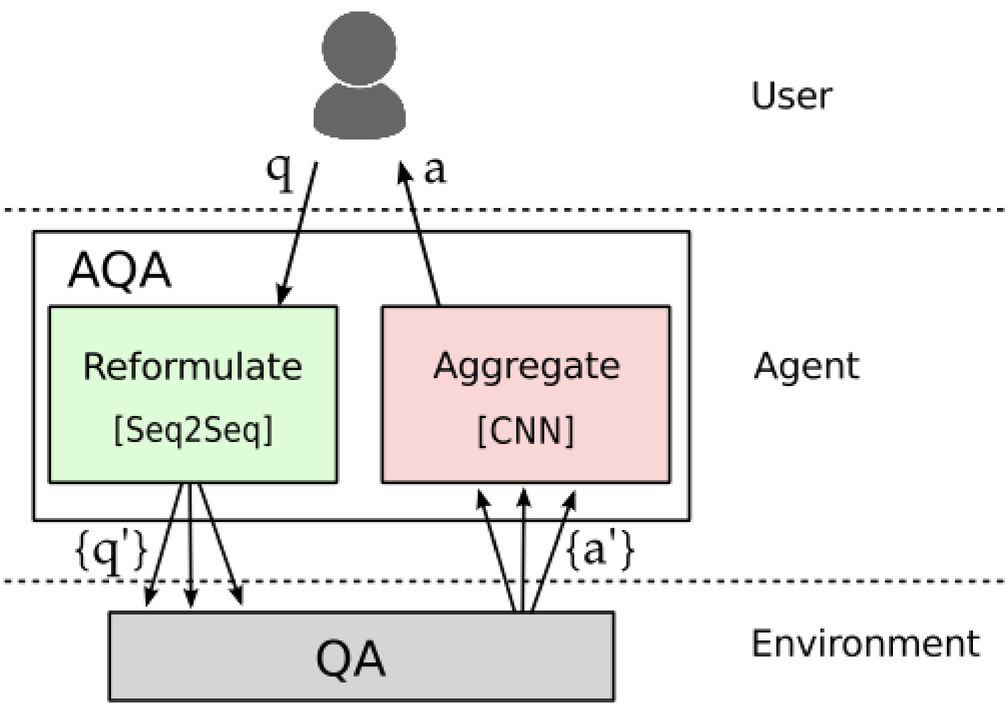
ASK THE RIGHT QUESTIONS: ACTIVE QUESTION REFORMULATION WITH REINFORCEMENT LEARNING

Created	@Jun 21, 2019 2:40 PM
Tags	
Updated	@Jul 01, 2019 5:32 PM

ACTIVE QUESTION ANSWERING M ODEL



QUESTION-ANSWERING ENVIRONMENT

- Use BiDirectional Attention Flow (BiDAF) as black-box environment

REFORMULATION MODEL

- a sequence-to-sequence model
- decoder reformulates utterances {q'} in the same language

ANSWER SELECTION MODEL

- selects the best answer from the set {a'}

TRAINING

QUESTION ANSWERING ENVIRONMENT

- BiDAF becomes the black-box environment and its parameters are not updated further
- The agent to learn to communicate using natural language with an environment over which is has no control

POLICY GRADIENT TRAINING OF THE REFORMULATION MODEL

- maximizing a reward a∗ = argmax a R(a|q0)
- R is the token level F1 score on the answer
- The policy is a sequence-to-sequence model

$$\pi_\theta(q|q_0) = \prod_{t=1}^{T} p(w_t|w_1, \ldots, w_{t-1}, q_0) \qquad (1)$$

- The goal is to maximize the expected reward of the answer
  - compute gradients for training using REINFORCE

$$\mathbb{E}_{q\sim\pi_\theta(\cdot|q_0)}[R(f(q))] \approx \frac{1}{N}\sum_{i=1}^{N} R(f(q_i)), \quad q_i \sim \pi_\theta(\,\cdot\,|q_0) \qquad (2)$$

  - compute an unbiased estimate with Monte Carlo sampling
  - θ are the policy’s parameters q ∼ πθ (· |q0) ?? notation의미 이애안됨

$$\nabla \mathbb{E}_{q\sim\pi_\theta(\cdot|q_0)}[R(f(q))] = \mathbb{E}_{q\sim\pi_\theta(\cdot|q_0)} \nabla_\theta \log(\pi_\theta(q|q_0))R(f(q)) \tag{3}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log(\pi(q_i|q_0))R(f(q_i)), \quad q_i \sim \pi_\theta(\,\cdot\,|q_0) \tag{4}$$

- collapse onto a sub-optimal deterministic policy. use entropy regularization

$$H[\pi_\theta(q|q_0)] = -\sum_{t=1}^{T}\sum_{w_t\in V} p_\theta(w_t|w_{<t},q_0)\log p_\theta(w_t|w_{<t},q_0) \tag{5}$$

- This final objective
  - B(q0): baseline reward
  - λ is the regularization weight

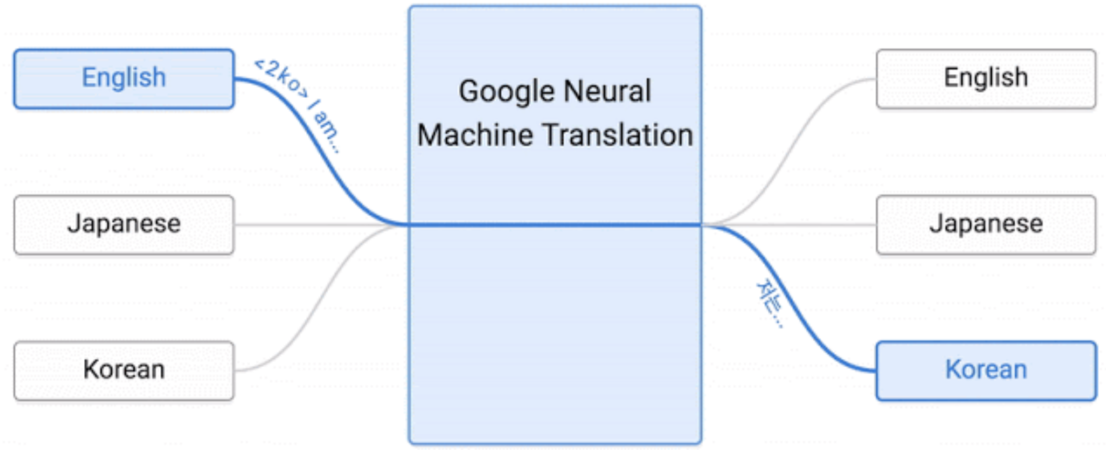
$$\mathbb{E}_{q\sim\pi_\theta(\cdot|q_0)}[R(f(q)) - B(q_0)] + \lambda H[\pi(q|q_0)], \tag{6}$$

ANSWER SELECTION

- generate (query, rewrite, answer) tuples
- train another neural network to pick the best answer from the candidates
- CNN which offers good computational efficiency and accuracy

PRE TRAINING OF THE REFORMULATION MODEL

- English-English corpora are scarce
- produce a multilingual translation system that translates between several languages
- zero-shot translation



EXPERIM ENTS

QUESTION ANSWERING DATA AND BIDAF TRAINING

- Dataset: SearchQA

QUESTION REFORMULATOR TRAINING

- United Nations Parallel Corpus (Arabic, English, Spanish, French, Russian, and Chinese)
    - train the zero-shot neural MT system
    - poor quality
- Paralex database of question paraphrases
    - refined model has visibly better quality than the zero-sho
- reinforcement-learning based tuning ???

TRAINING THE ANSWER SELECTOR

- generate N = 20 rewrites for each question in the SearchQA training and validation sets

BASELINES AND BENCHMARKS

- Attention Sum Reader (ASR)
- BiDAF to answer the original question

RESULTS

		Baseline		MI-SubQuery		Base-NMT		AQA				
		ASR	BiDAF	TopHyp	CNN	TopHyp	CNN	TopHyp	Voting	MaxConf	CNN	Human
Dev	EM	-	31.7	24.1	37.5	26.0	37.5	32.0	33.6	35.5	**40.5**	-
	F1	24.2	37.9	29.9	44.5	32.2	44.8	38.2	40.5	42.0	**47.4**	-
Test	EM	-	28.6	23.2	35.8	24.8	35.7	30.6	33.3	33.8	**38.7**	43.9
	F1	22.8	34.6	29.0	42.8	31.0	42.9	36.8	39.3	40.2	**45.6**	-

- MI-SubQuery: generates reformulation candidates by enumerating all subqueries of the original SearchQA query
- Base-NMT: the zero-shot monolingual NMT system trained without reinforcement learning
- TopHyp: use the top hypothesis generated by the sequence model
- Voting: use BiDAF scores for a heuristic weighted voting scheme

$$\operatorname{argmax}_a \sum_{a'=a} s(a')$$

- MaxConf: select the answer with the single highest BiDAF score
- CNN: complete system with the learned CNN model

SRC

- Environment
  - px/environment/bidaf.py
- Reformulation
  - px/nmt/mode.py
    - loss: _compute_loss_offset_and_advantages
- px/selector/selector_keras.py