Learning to Compose Neural Networks for Question Answering

Paper by J. Andreas, M. Rohrbach, T. Darrell & D. Klein, 2016

Presented by Kaifu Wang

February 19, 2020

Task: QA about World Representations

VQA: Q: `How many children are in the bed?`

2 1

GeoQA: Q: `How many states are in the United States?`

$$[\![\texttt{\#states(USA)}]\!]_{\mathcal{W}_{1776}} = 13 \quad [\![\texttt{\#states(USA)}]\!]_{\mathcal{W}_{2020}} = 50$$

Goal: build a model for $\mathbb{P}(y|x, w)$ that can answer questions x based on a variety of *world representations* w which encodes external knowledge and observations, such as knowledge bases and images.

Previous Approaches

Task: given question x and world representation w, predict $\mathbb{P}(y|x,w)$.

- Approach 1: Learning encodings for x and w, and build a classifier for $[x;w]$.
  - Hard to interpret the models.
  - Sentence embedding may not capture linguistic complexity.
- Approach 2: Map questions to logical forms. For example: question `Which state has the largest area` can be mapped to the following logical form of the world representation:

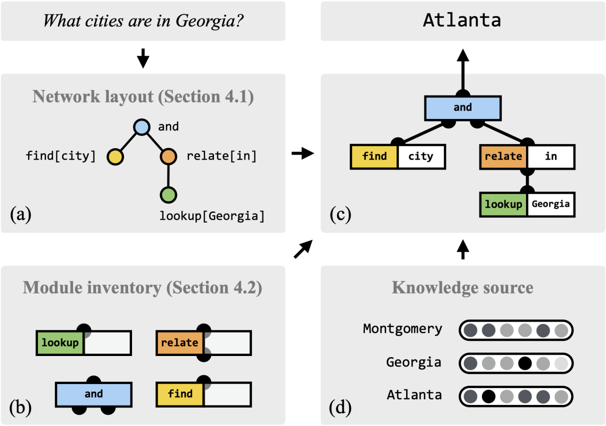
  `argmax(area(state())).`
  - It can be hard to deal with visual features in purely logical rules.
  - May require combinatorial optimization.

Model: Dynamic Neural Module Network

Question x

Net Layout z

Modules



Answer y

NN

World Rep w

Model: $x \to z \to y \leftarrow w$

Execution Model $\mathbb{P}(y|z, w; \theta_e)$

Given the layout z, the NN is assembled using the following modules:

Module	Input	Output	Example (**Arguments**, *inputs*)
`Lookup`	World	Attention	Find **"Georgia"** in the *database*
`Find`	World	Attention	Find the **bird** in the *picture*
`Relate`	Attention	Attention	Which cities are **in** *Pennsylvania*?
`And`	Attentions	Attention	Find **sheep's ear**
`Describe`	Attention	Labels	What is the **color**?
`Exists`	Attention	Labels	Are there any **cities**?

Arguments are provided by z, while *inputs* are the information flow of w. These modules are shallow NNs with trainable parameters. For example,

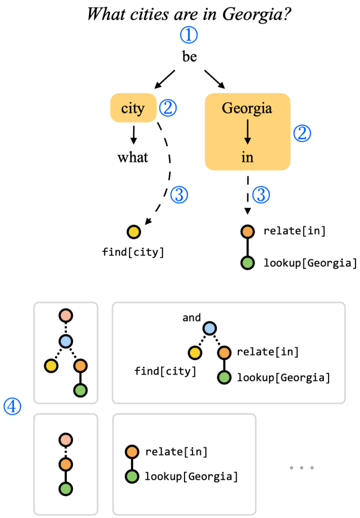
$$[\![\texttt{find}_{[\texttt{bird}]}]\!] = \mathrm{Softmax}(a \odot \sigma(Bv^{\texttt{bird}} \oplus CW \oplus d))$$

A reasonable parsing for question "`What color is the bird?`" can be

```
describe[color](find[bird]())
```

Candidate set of the layout $\{z_1, z_2 \ldots, z_n\}$ is constructed by:

1. Find dependency parse of x.
2. Collect all nouns, verbs, prepositional phrases attached directly to a wh-word or copula.
3. Map each of these words/phrases to a fragment of module(s) using hand-written rule.
4. For each subset of these fragments, construct a layout candidate by joining all modules with an `And` module, and insert a `Describe` or `Measure` module at the top.



Layout Model $\mathbb{P}(z|x;\theta_\ell)$

For $z \in \{z_1, \ldots, z_n\}$, $\mathbb{P}(z|x;\theta_\ell)$ is modeled as a shallow NN:

$$\mathbb{P}(z|x;\theta_\ell) = \mathrm{Softmax}(s(z|x))$$
$$\underbrace{s(z_i|x)}_{\substack{\text{Layout}\\ \text{Score}}} = a^\mathsf{T}\sigma(B\,\underbrace{h_q(x)}_{\substack{\text{Question}\\ \text{Embedding}}} + C\,\underbrace{f(z_i)}_{\substack{\text{Layout}\\ \text{Feature}}} + d)$$

Here, the layout feature vector includes indicators on the number of modules of each type present, and their associated parameter arguments.

The complete model $\mathbb{P}(y|x, w)$ is now well-defined. However,

- Computing the execution model $\mathbb{P}(y|z, w; \theta_e)$ is expensive.
- Computing the layout model $\mathbb{P}(z|x; \theta_\ell)$ is relatively cheaper.

Analogy to RL: viewing z as actions and the likelihood of y as reward.

1. Sample a layout z according to $\mathbb{P}(z|x; \theta_\ell)$.
2. Update θ_e by ordinary back-propagation.
3. Update θ_ℓ along the gradient of reward function, which is computed according to the REINFORCE rule:

$$\nabla J_k(\theta_\ell) = \mathbb{E}_y[(\nabla \log \mathbb{P}(z|x; \theta_\ell)) \cdot \underbrace{\log \mathbb{P}(y|z, w; \theta_e)}_{\text{reward}}]$$

Experiment: VQA

VQA tasks

To modeling prior knowledge of QA, add an additional layer:

$$\log \mathbb{P}(y|w,x) = (Ah_q(x) + B[\![z]\!]_w)_y$$

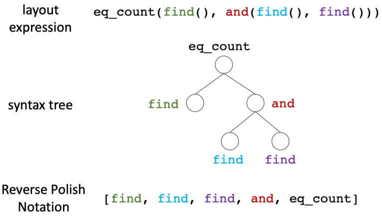
to allow direct influence of x on y.

	test-dev				test-std
	Yes/No	Number	Other	All	All
Zhou (2015)	76.6	35.0	42.6	55.7	55.9
Noh (2015)	80.7	37.2	41.7	57.2	57.4
NMN	77.7	37.2	39.3	54.8	55.1
NMN*	79.7	37.1	42.8	57.3	–
D-NMN	80.5	37.4	43.1	57.9	**58.0**

Results

N2NMN: R. Hu, et al., *Learning to Reason: End-to-End Module Networks for Visual Question Answering*. in ICCV, 2017.

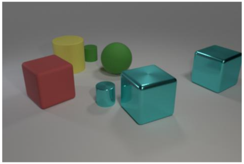
- Use *attentional* modules where hard-coded arguments are replaced by soft attention over question words v_i: $x^{(m)}_{\text{txt}} = \sum_{i=1} \alpha^{(m)}_i v_i$.
- Layout policy:
  1. Search in the space of all possible layouts, rather than in a finite candidate set.
  2. Express the layout as a linear sequence of module tokens using the *Reverse Polish Notation*, so the layout prediction becomes a seq2seq problem. One can use a encoder-decoder model for prediction.



- Apart from end-to-end training, pre-train the layout model by **behavioral cloning from expert polices**: use an expert polict $\mathbb{P}_e(z|x)$ to directly supervise and initialize the layout model.
- VQA experiment result:

Method	Visual feature	Accuracy
NMN [3]	LRCN VGG-16	57.3
D-NMN [2]	LRCN VGG-16	57.9
MCB [9]	ResNet-152	64.7
ours - cloning expert	LRCN VGG-16	61.9
ours - cloning expert	ResNet-152	64.2
ours - policy search after cloning	ResNet-152	**64.9**

- New datasets: SHAPES and CLEVR.



How many other things are of the same size as the green matte ball?

Other Applications

- NMN for QA: N. Gupta et al., *Neural Module Network for Reasoning over Text*. in ICLR, 2020.
- NMN for proving: T. Rocktaschel and S. Riedel, *End-to-End Differentiable Proving*. in NIPS, 2017.

Thank you!

Questions?