Learning to Compose Neural Networks for Question Answering (a.k.a. Dynamic Neural Module Networks)

Authors: Jacob Andreas, Marcus Rohrbach, Trevor Darrell, Dan Klein
Presented by: K.R. Zentner

Basic Outline

- Problem statement
- Brief review of Neural Module Networks
- New modules
- Learned layout predictor
- Some minor additions
- Results
- Conclusion

Problem Statement

Would like to have a single algorithm for a variety of question answering domains.

More precisely, given a question *q* and a world *w*, produce an answer *y*.

q is a natural language question, *y* is a label (or boolean), *w* can be visual **or semantic.**

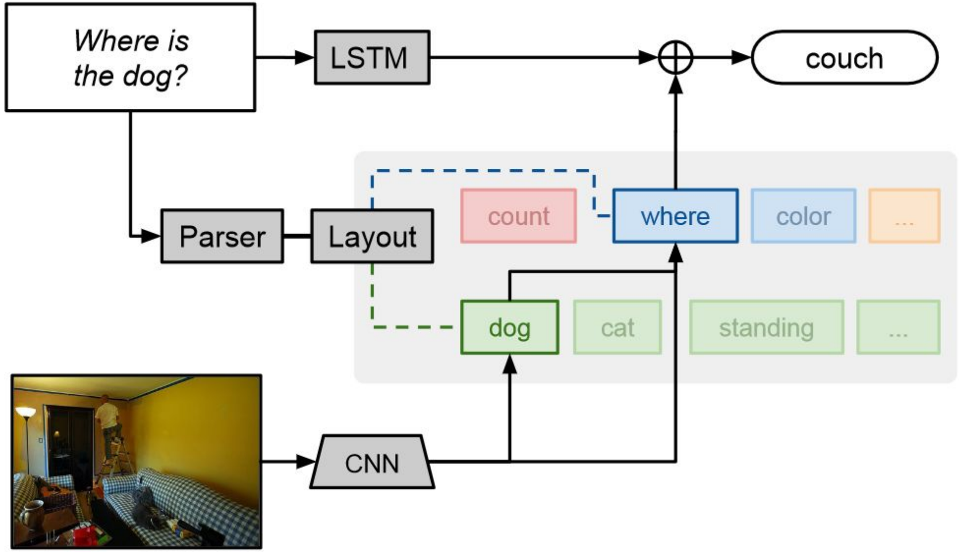
Would like to work well with a small amount of data, but still benefit from significant amounts of data.

Neural Module Networks

Answer a question over an input (image only), in two steps:

1. *Layout* a network from the question.

2. *Evaluate* the network on the input.



Neural Module Networks

Two large weaknesses:

1. What if we don't have an image as input?

2. What if dependency parsing results in a bad network layout?

What if we don’t have an image as input?

Replace Image with “World”

- The “World” is an arbitrary set of vectors.
- Still use attention across the vectors.
- Treat image as world by operating after the CNN.
- NMN modules assume CNN / Image!

New Modules!

Neural Module Network	Dynamic Neural Module Network
`attend[word]`: *Image → Attention*	`find[word]`: *(World) → Attention*
	`lookup[word]`: *() → Attention*
`re-attend[word]`: *Attention → Attention*	`relate[word]`: *(World) Attention → Attention*
`combine[word]`: *Attention x Att. → Attention*	`and`: *Attention* → Attention*
`classify[word]`: *Image x Attention → Label*	`describe[word]`: *(World) Attention → Labels*
`measure[word]`: *Attention → Label*	`exists`: *Attention → Labels*

Attend → Find

Neural Module Network	Dynamic Neural Module Network
`attend[word]`: *Image* → *Attention*	`find[word]`: *(World)* → *Attention*
A convolution.	“An MLP:” softmax($a \odot \sigma(Bv_i \oplus CW \oplus d)$)
`attend[dog]`	`find[dog]` or `find[city]`
Generates an attention over the *Image*.	Generates an attention over the *World*.

“ “ → Lookup

Neural Module Network	Dynamic Neural Module Network
	`lookup[word]`: *()* → *Attention*
	A know relation: $e_{f(i)}$
	`lookup[Georgia]`
	For words with constant attention vectors.

Re-attend → Relate

Neural Module Network	Dynamic Neural Module Network
`re-attend[word]`: *Attention → Attention*	`relate[word]`: *(World) Attention → Attention*
(FC → ReLU) x 2	softmax(a $\odot$ σ(B$v_i \oplus$ CW $\oplus$ Dw(h) $\oplus$ e))
`re-attend[above]`	`relate[above]` or `relate[in]`
Generates a new attention over the *Image*.	Generates a new attention over the *World*.

Combine → And

Neural Module Network	Dynamic Neural Module Network
`combine[word]`: *Attention x Att. → Attention*	`and`: *Attention* → Attention*
Stack → Conv. → ReLU	h1 ⊙ h2 ⊙ …
`combine[except]`	`and`
Combines two *Attentions* in an arbitrary way.	Multiplies attentions (analogous to set intersection).

Classify → Describe

Neural Module Network	Dynamic Neural Module Network
`classify[word]`: *Image x Attention → Label*	`describe[word]`: *(World) Attention → Labels*
Attend → FC → Softmax	softmax(Aσ(Bw(h) + vi))
`classify[where]`	`describe[color]` or `describe[where]`
Transforms an *Image* and *Attention* into a *Label.*	Transforms a *World* and *Attention* into a *Label.*

Measure → Exists

Neural Module Network	Dynamic Neural Module Network
`measure[word]`: *Attention* → *Label*	`exists`: *Attention* → *Labels*
FC→ ReLU → FC → Softmax	softmax((argmax h) a + b)
`measure[exists]`	`exists`
Transforms just an *Attention* into a *Label.*	Transforms just an *Attention* into a *Label.*

What if dependency parsing results in a bad network layout?

New layout algorithm!

NMN

- Dependency parse
  - Leaf → attend
  - Internal (arity 1) → re-attend
  - Internal (arity 2) → combine
  - Root (yes/no) → measure
  - Root (other) → classify
- Layout of network strictly follows structure of dependency parse tree.

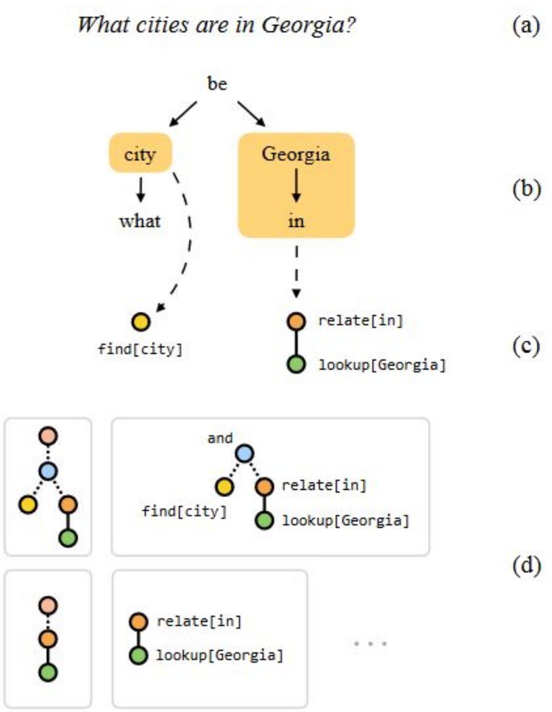
Dynamic-NMN

- Dependency parse
  - Proper nouns → lookup
  - Nouns & Verbs → find
  - Prepositional phrase → relate + find
- Generate candidate layouts from subsets of fragments.
  - and all fragments in subset
  - measure or combine
- “Rank” layouts with structure predictor.
- Use highly ranked layout.

New layout algorithm!

Only possible because “and” module has no parameters.

Structure predictor doesn’t have any direct supervision. How can we train it?



Structure Predictor?

Computes $h\_q(x)$ by passing LSTM over question.

Computes featurization $f(z\_i)$ of ith layout.

Sample layout with probability $p(z\_i \mid x; \theta\_l) = \mathrm{softmax}(a \cdot \sigma(B\ h\_q(x) + C\ f(z\_i) + d))$

How to train Structure Predictor?

Use a gradient estimate, as in REINFORCE (Williams, 1992).

Want to perform an SGD update with $\nabla J(\theta\_l)$.

Estimate $\nabla J(\theta\_l) = E[\nabla \log p(z \mid x ; \theta\_l) \cdot r]$

Use reward $r = \log p(y \mid z, w; \theta\_e)$

Step in direction $\nabla \log p(z \mid x ; \theta\_l) \cdot \log p(y \mid z, w; \theta\_e)$

With small enough learning rate, estimate should converge.

New Dataset: GeoQA (+ Q)

- Entirely semantic: database of relations.
- Very small: 263 examples.
- (+ Q) adds quantification questions (e.g. What cities are in Texas? → Are there any cities in Texas?)
- State of the art results.
    - Compared to 2013 baseline and NMN.

	Accuracy	
Model	GeoQA	GeoQA+Q
LSP-F	48	–
LSP-W	51	–
NMN	51.7	35.7
D-NMN	**54.3**	**42.9**

Table 2: Results on the GeoQA dataset, and the GeoQA dataset with quantification. Our approach outperforms both a purely logical model (LSP-F) and a model with learned perceptual predicates (LSP-W) on the original dataset, and a fixed-structure NMN under both evaluation conditions.

Old Dataset: VQA

- Need to add “passthrough” to final hidden layer.
- Once again uses pre-trained VGG network.
- Slightly improved state of the art.

	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
Yang (2015)	79.3	36.6	46.1	58.7	58.9
NMN	81.2	38.0	44.0	58.6	58.7
D-NMN	81.1	38.6	45.5	59.4	**59.4**

Table 1: Results on the VQA test server. NMN is the parameter-tying model from Andreas et al. (2015), and D-NMN is the model described in this paper.

Weaknesses?

- Can only generate very flat layouts, with only one conjunction or quantifier.
- Gradient estimate probably much more expensive / unstable than true gradient.
- Not any simpler than NMN, which are already considered complex.
- Similar in spirit but not implementation to Neural Symbolic VQA (Yi et. al. 2018).
- Much more complex than Relation Networks (Santoro et. al. 2017).

Questions? Discussion.