Learning Graphical State Transitions

Daniel D. Johnson | Harvey Mudd College | ddjohnson@hmc.edu

Motivation

Many different types of data can be formulated using a graph structure: *relationships between entities* map to *edges between nodes*. Existing work can take graph-structured data as input, but not produce or modify it. Goals of this research:

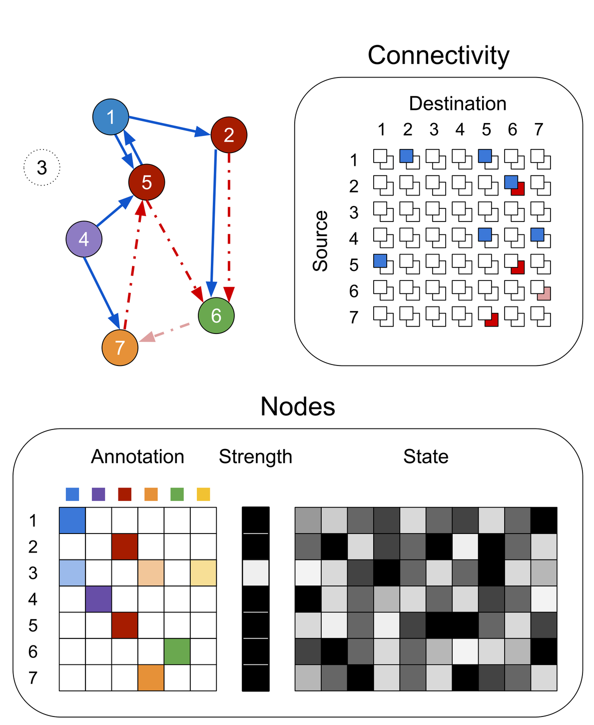
- Design a neural network architecture that can manipulate graphical states.
- Use this architecture to solve tasks with graphical internal state and/or graphical output.

Previous Work

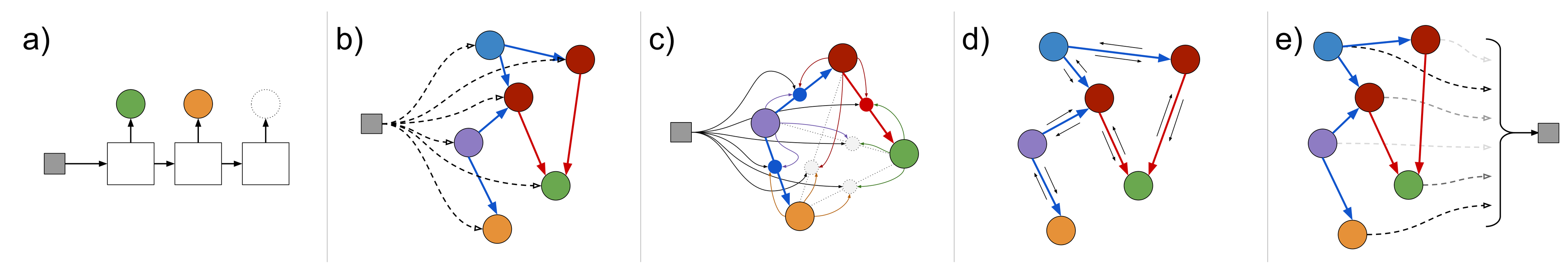
- *Graph Neural Network* (Gori et al., 2005; Scarselli et al., 2009): Each node has a state vector. States are updated based on adjacent nodes until convergence.
- *Gated Graph Neural Network* (Li et al., 2016): Like GNN, but compute states using fixed number of GRU-style updates, train with back-propagation.

Graph Representation

- Set of nodes $v \in \mathcal{V}$, each with:
  - strength s_v (with $0 \le s_v \le 1$)
  - annotation $\mathbf{x}_v \in \mathbb{R}^N$ where $\sum_{j=1}^{N} x_{v,j} = 1$
  - hidden state $\mathbf{h}_v \in \mathbb{R}^D$
- Connectivity matrix $\mathcal{C} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times Y}$
  - $\mathcal{C}_{v,v',y}$: strength of directed edge of type y from v to v' (with $0 \le \mathcal{C}_{v,v',y} \le 1$)



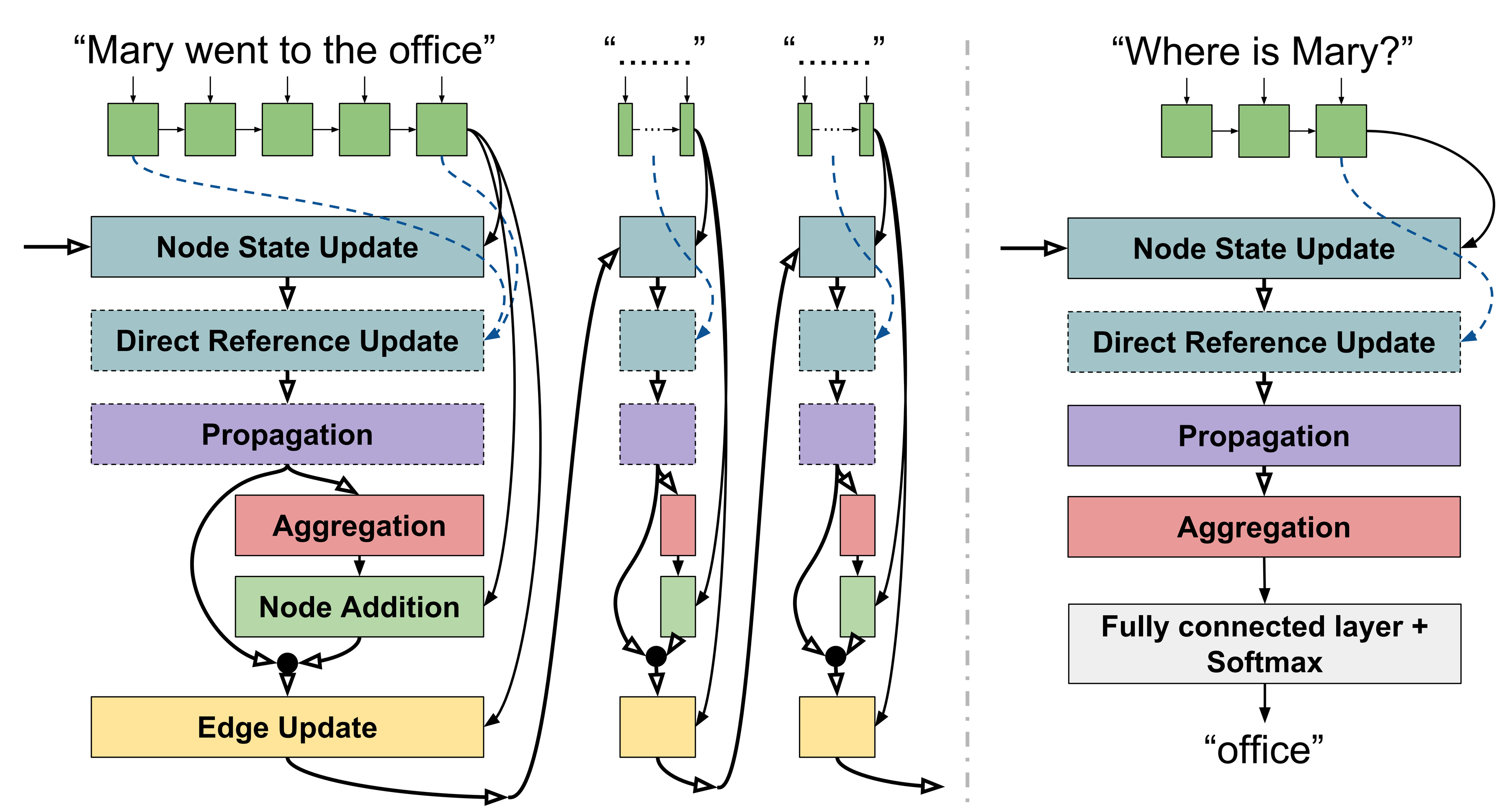
Graph Transformations



a) Node addition: create new nodes
b) Node state update: update node states based on new input
   Direct reference update: update specific node types mentioned in input
c) Edge update: add or remove edges based on node states
d) Propagation: update node states based on adjacent node states
e) Aggregation: combine node states into a representation vector

Gated Graph Transformer Neural Network

By combining graph transformations, the GGT-NN can solve question-answering and graph construction tasks.



Transformations are modular and can be removed or reconfigured to solve different task types.

Training the GGT-NN

During training, the correct graph states are substituted in after each input sentence. The GGT-NN is trained to reproduce these graph states, and answer the final query correctly using the last graph state. In addition to the loss for answering the query (which depends on the task), the network is trained to minimize:

Edge loss: $\mathcal{L}_{\text{edge}} = -\mathcal{C}^* \cdot \ln(\mathcal{C}) - (1-\mathcal{C}^*)\cdot\ln(1-\mathcal{C})$

Node loss: $\mathcal{L}_{\text{node}} = -\max_{\pi} \sum_{v\in\mathcal{V}_{\text{added}}} s^*_{\pi(v)}\ln(s_v) + (1-s^*_{\pi(v)})\ln(1-s_v) + \mathbf{x}^*_{\pi(v)}\cdot\ln(\mathbf{x}_v)$

where π is the ordering of nodes that minimizes the loss, $*$ denotes the correct graph state, and $\cdot$ denotes the dot product.

bAbI Tasks

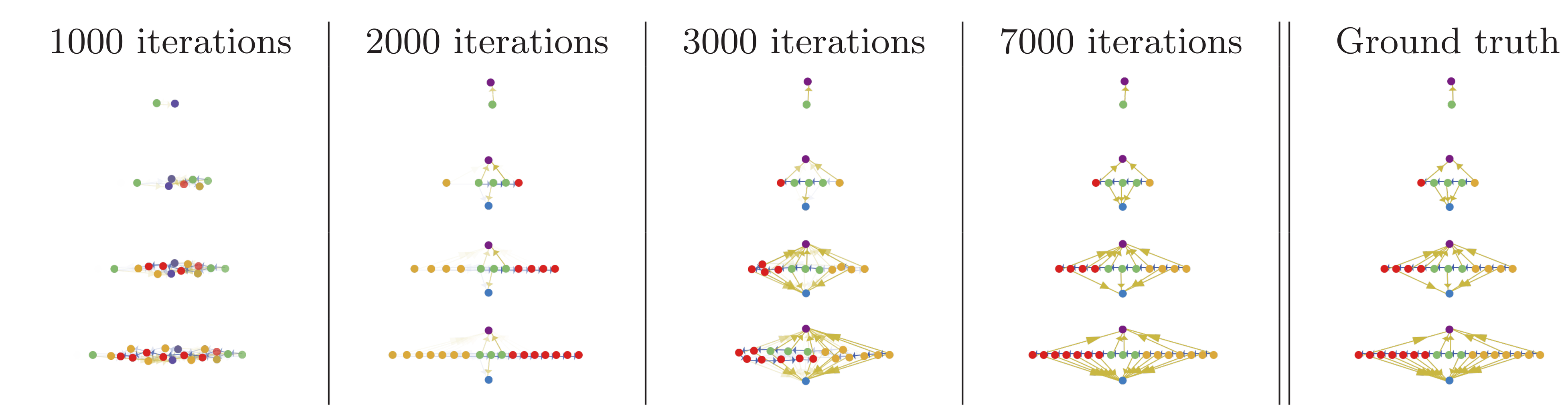
The GGT-NN model was evaluated on a dataset of simple synthetic question-answering tasks (Weston et al., 2016), and performance was compared to other models. Results:

	Examples needed: ≤ 5% error		% Error with 1,000 examples						% Error with 10,000 examples					
Task	GGT-NN + direct ref	GGT-NN	GGT-NN + direct ref	GGT-NN	LSTM	MemNN	MemN2N	EntNet	NTM	D-NTM	MemN2N*	DNC	DMN+	EntNet
1 - Single Supporting Fact	100	1000	**0**	**0.7**	50.0	**0**	**0**	**0.7**	31.5	**4.4**	**0**	**0**	**0**	**0**
2 - Two Supporting Facts	250	-	**0**	5.7	80.0	**0**	8.3	56.4	54.5	27.5	**0.3**	**0.4**	**0.3**	**0.1**
3 - Three Supporting Facts	1000	-	**1.3**	12.0	80.0	**0**	40.3	69.7	43.9	71.3	**2.1**	**1.8**	**1.1**	**4.1**
4 - Two Arg. Relations	1000	1000	**1.2**	**2.2**	39.0	**0**	**2.8**	**1.4**	**0**	**0**	**0**	**0**	**0**	**0**
5 - Three Arg. Relations	500	-	**1.6**	10.9	30.0	**2.0**	13.1	**4.6**	**0.8**	**1.7**	**0.8**	**0.8**	**0.5**	**0.3**
6 - Yes/No Questions	100	-	**0**	7.7	52.0	**0**	7.6	30.0	17.1	**1.5**	**0.1**	**0**	**0**	**0.2**
7 - Counting	250	-	**0**	5.6	51.0	15.0	17.3	22.3	17.8	6.0	**2.0**	**0.6**	**2.4**	**0**
8 - Lists/Sets	250	1000	**0**	**3.3**	55.0	9.0	10.0	19.2	13.8	**1.7**	**0.9**	**0.3**	**0**	**0.5**
9 - Simple Negation	250	-	**0**	11.6	36.0	**0**	13.2	31.5	16.4	**0.6**	**0.3**	**0.2**	**0**	**0.1**
10 - Indefinite Knowledge	1000	-	**3.4**	28.6	56.0	**2.0**	15.1	15.6	16.6	19.8	**0**	**0.2**	**0**	**0.6**
11 - Basic Coreference	100	1000	**0**	**0.2**	28.0	**0**	**0.9**	8.0	15.2	**0**	**0**	**0**	**0**	**0.3**
12 - Conjunction	500	1000	**0.1**	**0.7**	26.0	**0**	**0.2**	**0.8**	8.9	6.2	**0**	**0**	**0.2**	**0**
13 - Compound Coref.	100	1000	**0**	**0.8**	6.0	**0**	**0.4**	9.0	7.4	7.5	**0**	**0**	**0**	**1.3**
14 - Time Reasoning	1000	-	**2.2**	55.1	73.0	**1.0**	**1.7**	62.9	24.2	17.5	**0.2**	**0.4**	**0.2**	**0**
15 - Basic Deduction	500	500	**0.9**	**0**	79.0	**0**	**0**	57.8	47.0	**0**	**0**	**0**	**0**	**0**
16 - Basic Induction	100	500	**0**	**0**	77.0	**0**	**1.3**	53.2	53.6	49.6	51.8	55.1	45.3	**0.2**
17 - Positional Reasoning	-	-	34.5	48.0	49.0	35.0	51.0	46.4	25.5	**1.2**	18.6	12.0	**4.2**	**0.5**
18 - Size Reasoning	1000	-	**2.1**	10.6	48.0	**5.0**	11.1	8.8	**2.2**	**0.2**	5.3	**0.8**	**2.1**	**0.3**
19 - Path Finding	500	-	**0**	70.6	92.0	64.0	82.8	90.4	**4.3**	39.5	**2.3**	**3.9**	**0**	**2.3**
20 - Agent's Motivations	250	250	**0**	**1.0**	9.0	**0**	**0**	**2.6**	**1.5**	**0**	**0**	**0**	**0**	**0**

Graph Prediction Tasks

The GGT-NN was applied to two graph-prediction tasks: modeling the Rule 30 cellular automaton (Wolfram, 2002) and a 2-symbol 4-state universal Turing machine. Results on the original task (7 steps for the automaton, 6 steps for Turing machine) and on extended inputs of length 20 and 30 (to test generalization ability):

	Original	Length 20	Length 30
Cellular Automaton	100.0%	87.0%	69.5%
Turing Machine	99.9%	90.4%	80.4%



Conclusions

- GGT-NN model can construct and manipulate graphical state
- Modular graph transformations can be recombined in different ways
- Potentially useful for a wide variety of structured data applications

Future work:

- Reducing model supervision
- Sparse connectivity optimizations
- Extending node types

References

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*, 2016.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *ICLR*, 2016.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *ICLR*, 2016.

Stephen Wolfram. *A new kind of science*, volume 5. Wolfram media Champaign, 2002.