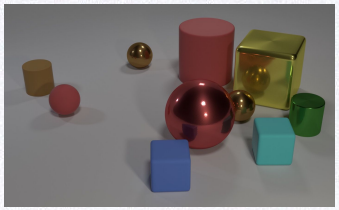
Learning Deep Generative Models of Graphs

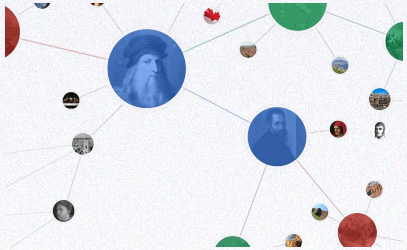
Yujia Li

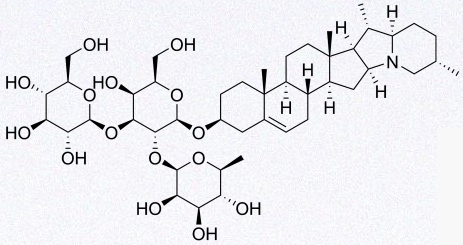
Collaborators: Oriol Vinyals, Chris Dyer, Razvan Pascanu, Peter Battaglia

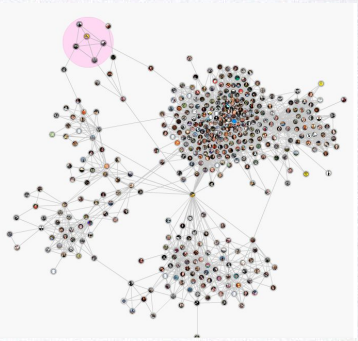
https://arxiv.org/abs/1803.03324

Graphs are Common Representations of Structure



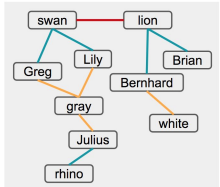






Context:

1. Lily is a swan.
2. Bernhard is a lion.
3. Greg is a swan.
4. Bernhard is white.
5. Brian is a lion.
6. Lily is gray.
7. Julius is a rhino.
8. Julius is gray.
9. Greg is gray.
10. Swan is afraid of lion.



Q: Is Greg afraid of Bernhard?
A: Yes.

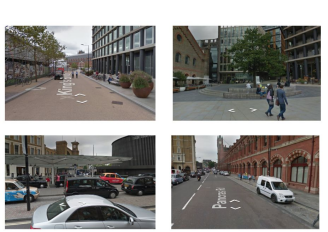
Q: Do Greg and Lily have the same color?
A: Yes.

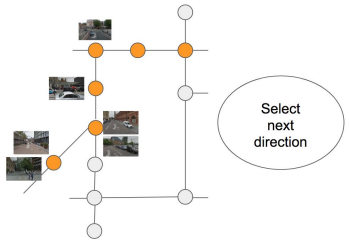
Q: Is Julius a lion?
A: No.

...

Infer graph

Use graph





Infer graph

Use graph

Why Generative Models of Graphs?

Infer graphs from input (map building, relation extraction, etc.)

Graph completion (knowledge graph completion, social networks)

Generate new structures (drug discovery)

Generative Models of Graphs

Stochastic Graph models

- Erdos-Renyi model, Barabasi-Albert model, stochastic block model, small-world model
- Nice theory, but limited capacity

Tree models

- Tons of tree generation models
- Only works on trees

Graph grammars

- Makes hard distinction between what is in the language vs not, hard to use

A Very General Graph Generation Process

loop until not adding new nodes:

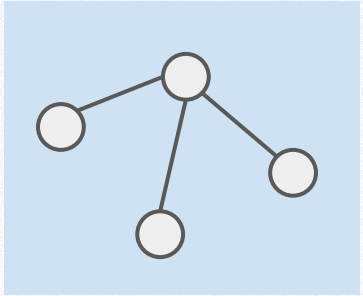
(1) add node?

(2) create node, add to the graph

(3) loop until not adding new edges:

  (a) add edge?

  (b) choose an existing node (and edge type) and create edge



No more nodes or a node type

A Very General Graph Generation Process

loop until not adding new nodes:

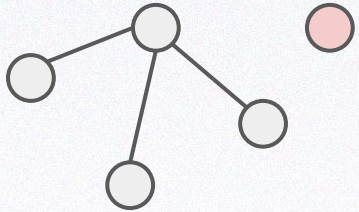
(1) add node?

(2) create node, add to the graph

(3) loop until not adding new edges:

  (a) add edge?

  (b) choose an existing node (and edge type) and create edge



A Very General Graph Generation Process

loop until not adding new nodes:

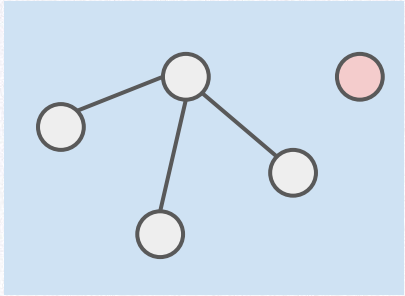
(1) add node?

(2) create node, add to the graph

(3) loop until not adding new edges:

(a) add edge?

(b) choose an existing node (and edge type) and create edge



{yes, no}

A Very General Graph Generation Process

loop until not adding new nodes:

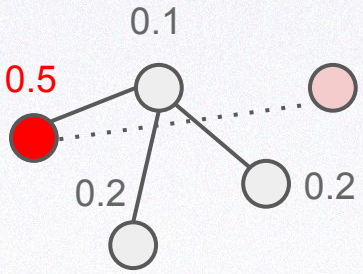
(1) add node?

(2) create node, add to the graph

(3) loop until not adding new edges:

(a) add edge?

(b) choose an existing node (and edge type) to create edge

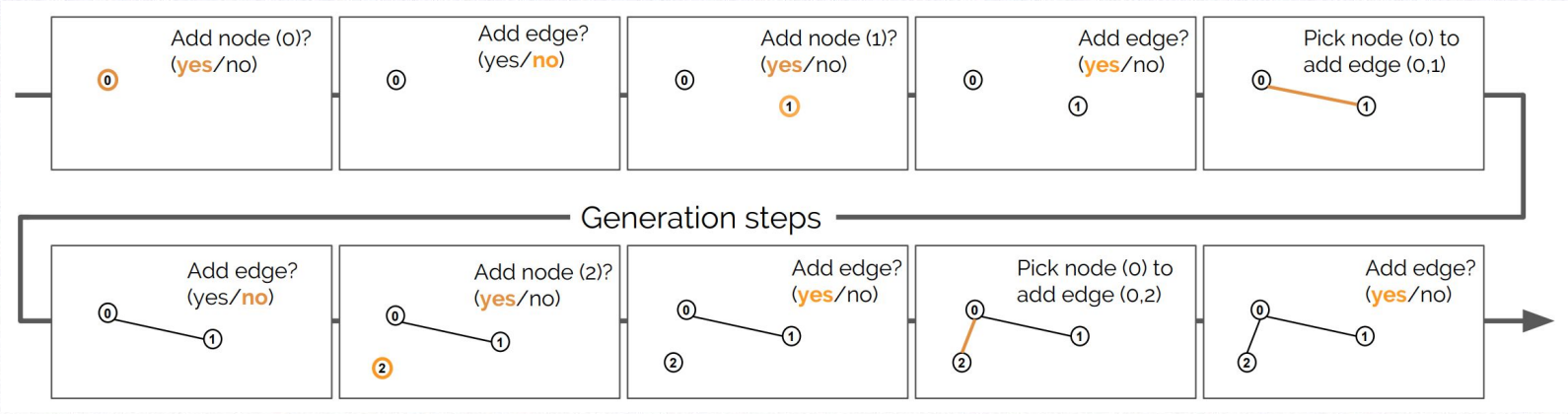


A Very General Graph Generation Process

No assumptions on the graph structure, capable of generating arbitrary graphs.

- Can handle typed / untyped, directed / undirected graphs easily.

Generation of a graph ⇒ A sequence of graph generating decisions.

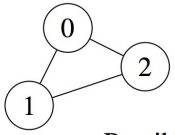


A Very General Graph Generation Process

3 Types of Decisions

- Add node (node type) vs not / Add edge vs not / Choose node (and edge type)

Order matters.



Possible Sequence 1:

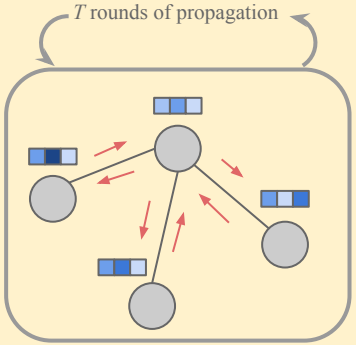
```
<add node (node 0)>
  <don't add edge>
<add node (node 1)>
  <add edge>
  <pick node 0 (edge (0, 1))>
  <don't add edge>
<add node (node 2)>
  <add edge>
  <pick node 0 (edge (0, 2))>
  <add edge>
  <pick node 1 (edge (1, 2))>
  <don't add edge>
<don't add node>
```

Possible Sequence 2:

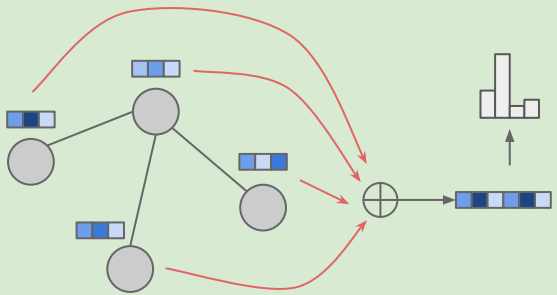
```
<add node (node 1)>
  <don't add edge>
<add node (node 0)>
  <add edge>
  <pick node 1 (edge (0, 1))>
  <don't add edge>
<add node (node 2)>
  <add edge>
  <pick node 1 (edge (1, 2))>
  <add edge>
  <pick node 0 (edge (0, 2))>
  <don't add edge>
<don't add node>
```

Model Details

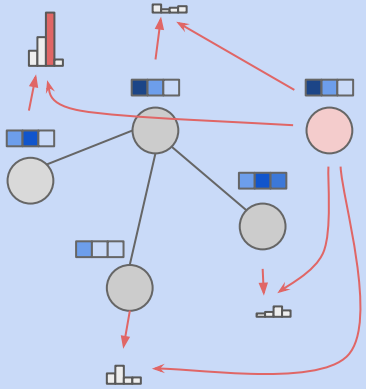
Compute graph representations by T rounds of propagation



Graph level predictions (add node / add edge)



Node selection (edge type selection)



Model Details

Each step: take a graph as input, make a prediction for that step.

Node states persist across steps, i.e. this is a recurrent model.

Easy to make it a conditional model

- Simply add a conditioning vector to the input, can go into any of the three modules or be used for node initialization.

Learning and Evaluation

Our model defines a joint distribution p(G, π) over graph G and its generation ordering π.

The marginal p(G) = Σ_{π} p(G, π) is intractable (sum over O(n!) terms).

We can do importance sampling

$$p(G) = \sum_{\pi} p(G, \pi) = \sum_{\pi} q(\pi \mid G) \frac{p(G, \pi)}{q(\pi \mid G)} = \mathbb{E}_{q(\pi|G)} \left[\frac{p(G, \pi)}{q(\pi \mid G)} \right]$$

Variance is minimized when q(π|G) = p(π|G).

Learning and Evaluation

Training to optimize marginal p(G) is intractable.

We learn the joint distribution instead by maximizing

$$\mathbb{E}_{p_{data}(G,\pi)}[\log p(G,\pi)] = \mathbb{E}_{p_{data}(G)}\mathbb{E}_{p_{data}(\pi|G)}[\log p(G,\pi)]$$

We pick $p_{data}(\pi|G)$, then take $q(\pi|G) = p_{data}(\pi|G)$ to match the evaluation process.

- Once the model is trained to optimum, $q(\pi|G) = p_{data}(\pi|G)$ gives the lowest variance estimator of the marginal likelihood.

Experiments

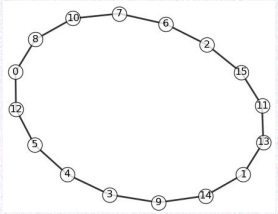
In the experiments we compare:

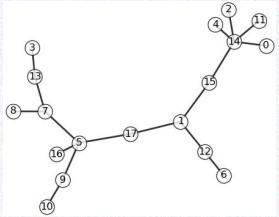
- Graph model vs. LSTMs
- Graph generating sequence vs. domain specific sequentialization

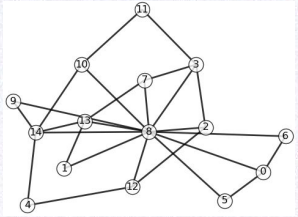
Metrics:

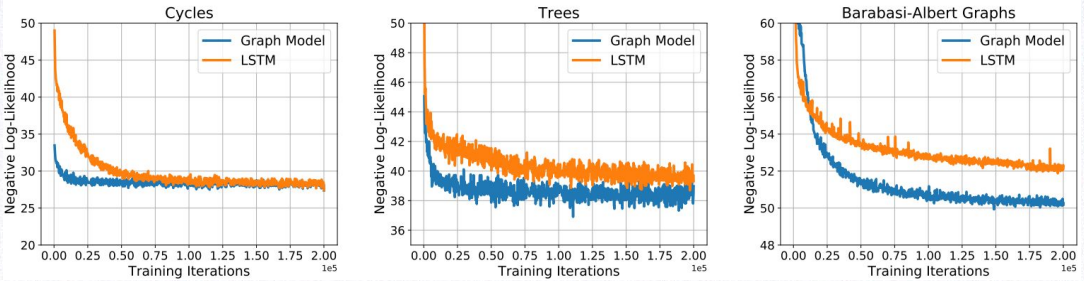
- log-likelihood
- other sample quality metrics when available

Generating Synthetic Graphs









Evaluating log p(G, π)
π ~ Uniform

Other sample quality measures

Dataset	Graph Model	LSTM	E–R Model
Cycles	**84.4%**	48.5%	0.0%
Trees	**96.6%**	30.2%	0.3%
B–A Graphs	**0.0013**	0.0537	0.3715

Molecule Generation

Data: sets of drug-like molecules, typed nodes, typed edges.

Arch	Grammar	Ordering	$\bar{N}$	%valid	%novel	Fixed	Best	Marginal
LSTM	SMILES	Fixed	1	93.59	81.27	17.28	15.98	15.90
LSTM	SMILES	Random	< 100	93.48	83.95	**15.95**	15.76	15.67
LSTM	Graph	Fixed	1	85.16	80.14	16.79	16.35	16.26
LSTM	Graph	Random	$O(n!)$	91.44	91.26	20.57	18.90	15.96
Graph	Graph	Fixed	1	**97.52**	90.01	16.19	**15.75**	15.64
Graph	Graph	Random	$O(n!)$	95.98	**95.54**	20.18	18.56	**15.32**

sample quality measure

evaluating likelihood on small molecules

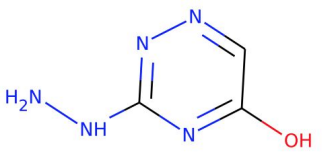


Figure 5. `NNc1nncc(O)n1`

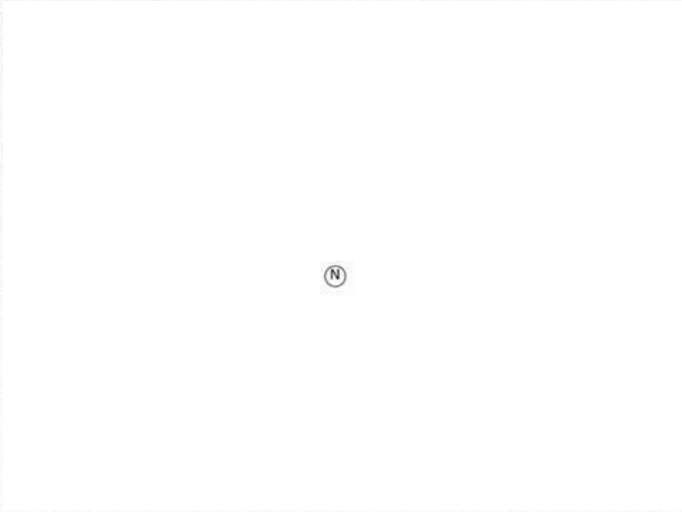
Comparing with previous methods (on another dataset):

- CVAE: close to 0% valid
- GrammarVAE: 34.9% valid
- Graph VAE: 13.5% valid
- **Ours: 89% valid and novel.**

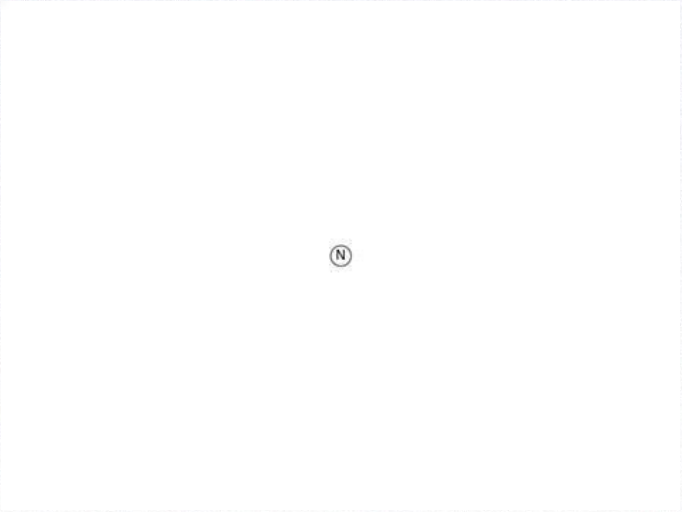
Molecule Generation

Different generation behavior when trained with different ordering

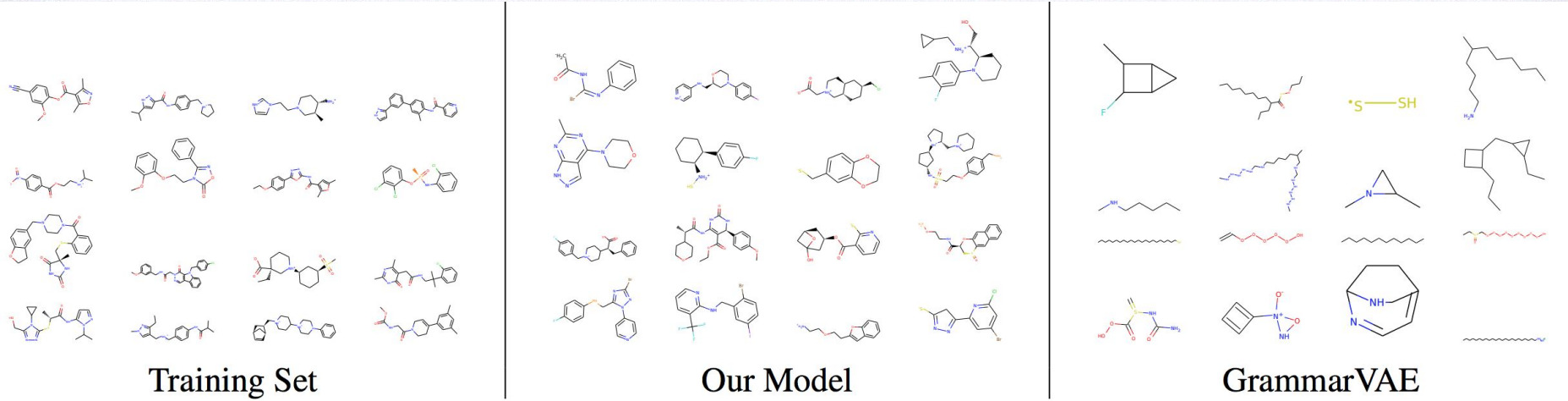
Canonical ordering



Uniform random ordering



Molecule Samples



Conditional Molecule Generation

Generate molecules conditioned on molecule properties

- Number of atoms, number of bonds, number of rings

Train on molecules with 0, 1, and 3 rings, test on molecules with

- 0 / 1 / 3 rings (same as training)
- 2 rings (interpolation)
- 4 rings (extrapolation)

					% property match			
Arch	Grammar	Condition	Valid	Novel	Atom	Bond	Ring	All
LSTM	SMILES	Training	84.3	82.8	71.3	70.9	**82.7**	**69.8**
LSTM	Graph	Training	65.6	64.9	63.3	62.7	50.3	48.2
Graph	Graph	Training	**93.1**	**92.1**	**81.7**	**79.6**	76.4	66.3
LSTM	SMILES	2-rings	64.4	61.2	7.1	4.2	43.8	0.5
LSTM	Graph	2-rings	54.9	54.2	23.5	21.7	23.9	9.8
Graph	Graph	2-rings	**91.5**	**91.3**	**75.8**	**72.4**	**62.1**	**50.2**
LSTM	SMILES	4-rings	71.7	69.4	46.5	3.7	1.3	0.0
LSTM	Graph	4-rings	42.9	42.1	16.4	10.1	3.4	1.8
Graph	Graph	4-rings	**84.8**	**84.0**	**48.7**	**40.9**	**17.0**	**13.3**

Generating Parse Trees and AMR Graphs (preliminary)

Parse tree generation (WSJ)

Model	Gen.Seq	Ordering	Perplexity	%Correct
LSTM	Tree	Depth-First	**1.114**	**31.1**
LSTM	Tree	Breadth-First	1.187	28.3
LSTM	Graph	Depth-First	1.158	26.2
LSTM	Graph	Breadth-First	1.399	0.0
Graph	Graph	Depth-First	1.124	28.7
Graph	Graph	Breadth-First	1.238	21.5

AMR graph generation

- Briefly tried for 1 week before ddl, results not good.

Future Directions

Ordering

- can we learn the ordering?

Long sequences

- the current graph generating sequence is quite long.

Scalability

- challenging to scale this approach to large graphs.

Difficulty in training

- training is unstable, so has to use very small learning rate - slow learning.