Graph Matching Networks for Learning the Similarity of Graph Structured Objects

Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, Pushmeet Kohli

DeepMind Google

Introduction

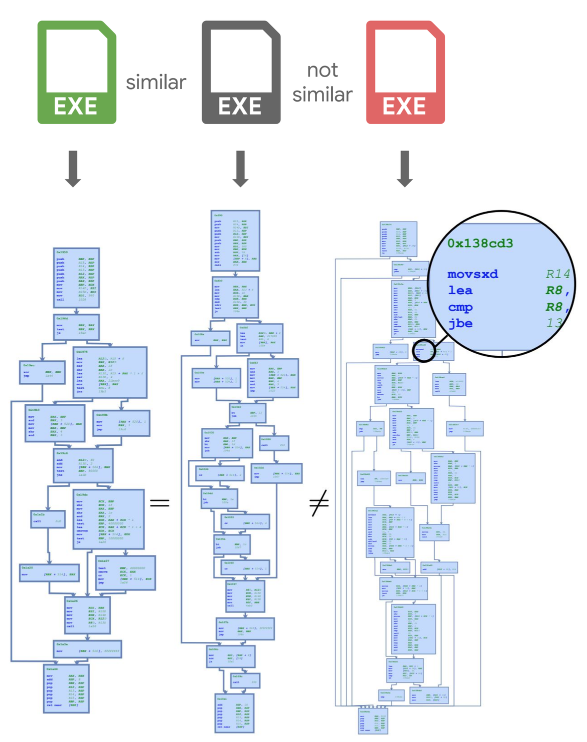
We want to learn a similarity / distance function between graphs.

Many applications:
- similarity search in graph databases
- copy detection for graph structured objects

Motivating problem:
- binary function similarity search for detecting software vulnerabilities

Challenge:
- reasoning about both graph structure as well as the graph semantics

Previous approaches:
- graph hashes
- graph kernels



Synthetic Task: Learning Graph Edit Distance

Learn a similarity metric that correlates with graph edit distance.
- Extreme case: distinguishing graph edit distance of 0 vs non-zero - graph isomorphism test.
- graph edit distance is NP-hard in general.

Comparing graph matching model vs graph embedding model vs WL-kernel on random graphs to distinguish edit distance of 1 vs 2.

Graph Spec.	WL kernel	embedding model	matching model
$n=20, p=0.2$	80.8 / 83.2	88.8 / 94.0	**95.0** / **95.6**
$n=20, p=0.5$	74.5 / 78.0	92.1 / 93.4	**96.6** / **98.0**
$n=50, p=0.2$	93.9 / **97.8**	95.9 / 97.2	**97.4** / 97.6
$n=50, p=0.5$	82.3 / 89.0	88.5 / 91.0	**93.8** / **92.6**

Measuring pair classification AUC / triplet accuracy.

Real World Tasks

Task 1: Binary function similarity search

Data:
- open source tool **ffmpeg** compiled into binary with different compilers (gcc / llvm / msvc++) and different optimization levels
- extract control flow graph from the binaries for each function
- binaries for the same function ⟶ similar

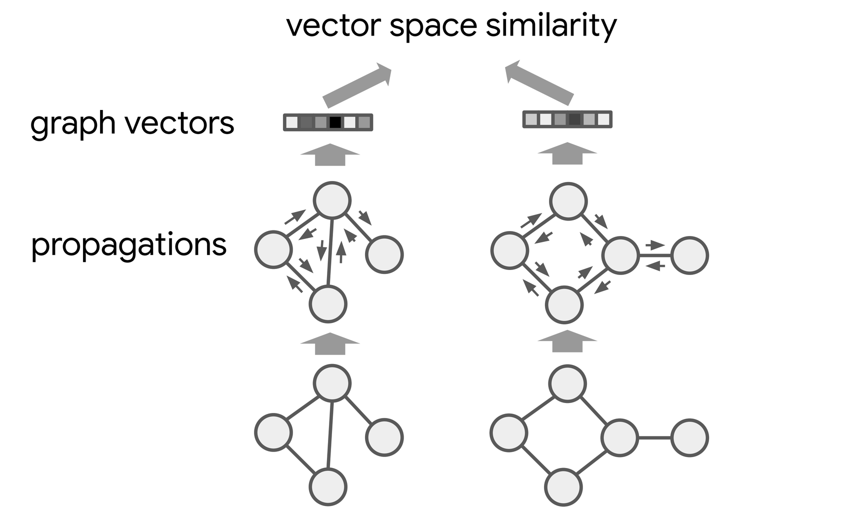
Baseline:
- a hand designed graph hash + learned LSH, used in a Google security project

Extra baselines:
- GCN: the graph convolutional networks
- Siamese networks: a Siamese version of the embedding models

The Models

GNN embedding model

Map each graph to a vector representation, through a graph neural net using multiple message passing / graph convolution layers.

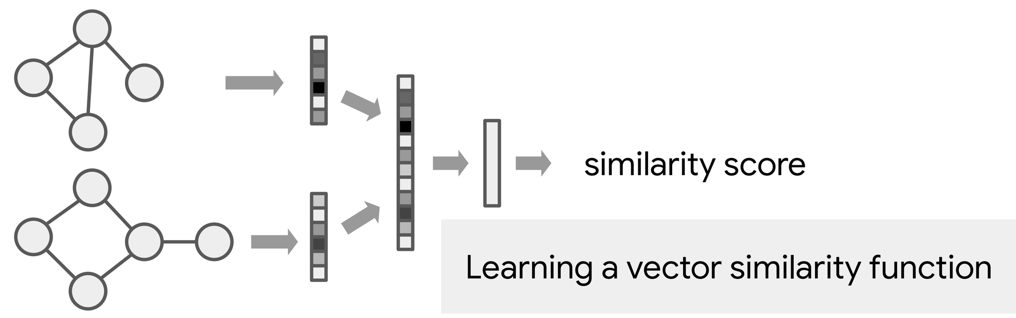


$\mathbf{x}_i$: node features, $\mathbf{x}_{ij}$: edge features, $\mathbf{h}_G$: graph vector

$$\mathbf{h}_i^{(0)} = \mathrm{MLP}_{\mathrm{node}}(\mathbf{x}_i), \quad \forall i \in V$$
$$\mathbf{e}_{ij} = \mathrm{MLP}_{\mathrm{edge}}(\mathbf{x}_{ij}), \quad \forall (i,j) \in E$$
$$\mathbf{m}_{j\to i} = f_{\mathrm{message}}(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e}_{ij}),$$
$$\mathbf{h}_i^{(t+1)} = f_{\mathrm{node}}\left(\mathbf{h}_i^{(t)}, \sum_{j:(j,i)\in E} \mathbf{m}_{j\to i}\right)$$
$$\mathbf{h}_G = f_G(\{\mathbf{h}_i^{(T)}\}) = \mathrm{MLP}_G\left(\sum_{i\in V} \sigma(\mathrm{MLP}_{\mathrm{gate}}(\mathbf{h}_i^{(T)})) \odot \mathrm{MLP}(\mathbf{h}_i^{(T)})\right)$$

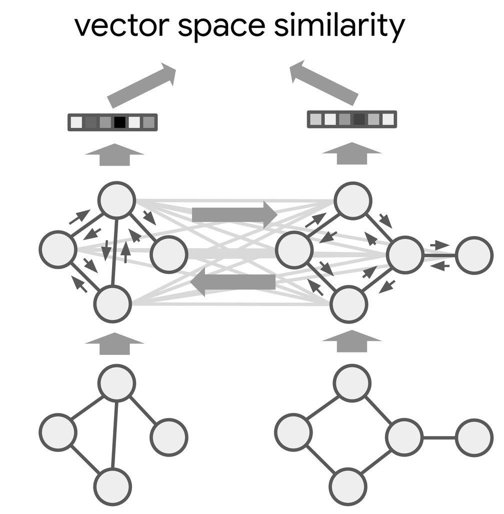
Graph similarity computed using standard metrics (Euclidean, Hamming etc.) in the vector space.

Siamese models



Graph matching networks

Cross-graph attention & comparison early in the message passing process.



$$\mathbf{m}_{j\to i} = f_{\mathrm{message}}(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e}_{ij}), \quad \forall (i,j) \in E_1 \cup E_2$$
$$\boldsymbol{\mu}_{j\to i} = f_{\mathrm{match}}(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}), \quad \forall i \in V_1, j \in V_2, \text{or } i \in V_2, j \in V_1$$
$$\mathbf{h}_i^{(t+1)} = f_{\mathrm{node}}\left(\mathbf{h}_i^{(t)}, \sum_j \mathbf{m}_{j\to i}, \sum_{j'} \boldsymbol{\mu}_{j'\to i}\right)$$
$$\mathbf{h}_{G_1} = f_G(\{\mathbf{h}_i^{(T)}\}_{i\in V_1}), \quad \mathbf{h}_{G_2} = f_G(\{\mathbf{h}_i^{(T)}\}_{i\in V_2}).$$

Cross-graph attention-based matching:

$$a_{j\to i} = \frac{\exp(s_h(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}))}{\sum_{j'} \exp(s_h(\mathbf{h}_i^{(t)}, \mathbf{h}_{j'}^{(t)}))}, \qquad \boldsymbol{\mu}_{j\to i} = a_{j\to i}(\mathbf{h}_i^{(t)} - \mathbf{h}_j^{(t)})$$
$$\text{so} \quad \sum_j \boldsymbol{\mu}_{j\to i} = \sum_j a_{j\to i}(\mathbf{h}_i^{(t)} - \mathbf{h}_j^{(t)}) = \mathbf{h}_i^{(t)} - \sum_j a_{j\to i}\mathbf{h}_j^{(t)}.$$

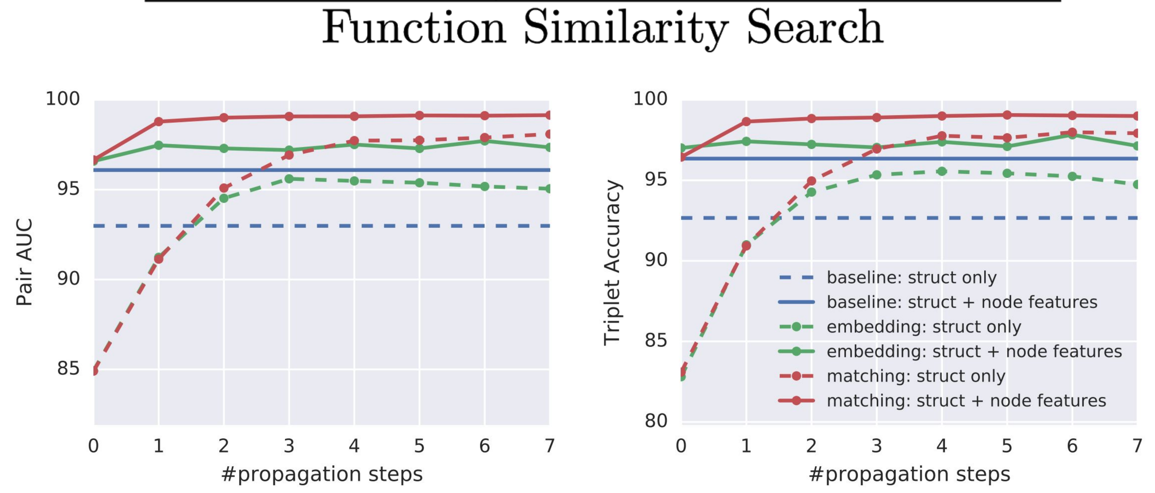
μ measures the difference between a node's vector with the closest nodes in the other graph, and is 0 when node vectors are identical.

Model	Pair AUC	Triplet Acc
Baseline	96.09	96.35
GCN	96.67	96.57
Siamese-GCN	97.54	97.51
GNN	97.71	97.83
Siamese-GNN	97.76	97.58
GMN	**99.28**	**99.18**

Function Similarity Search

Model	Pair AUC	Triplet Acc
GCN	94.80	94.95
Siamese-GCN	95.90	96.10
GNN	98.58	98.70
Siamese-GNN	98.76	98.55
GMN	**98.97**	**98.80**

COIL-DEL



More message passing ⟶ better performance

Task 2: Mesh graph similarity learning

Data:
- COIL-DEL dataset of mesh graphs.
- 100 object classes, each example is a graph
- same object class ⟶ similar

Learning

Learn to make similar pairs have small distance (high similarity), and dissimilar pairs have high distance (small similarity).

Pairwise training:

$$L_{\mathrm{pair}} = \mathbb{E}_{(G_1,G_2,t)}[\max\{0, \gamma - t(1 - d(G_1,G_2))\}]$$

$t \in \{-1,+1\}$: label, $+1$ for similar, otherwise -1. γ : margin

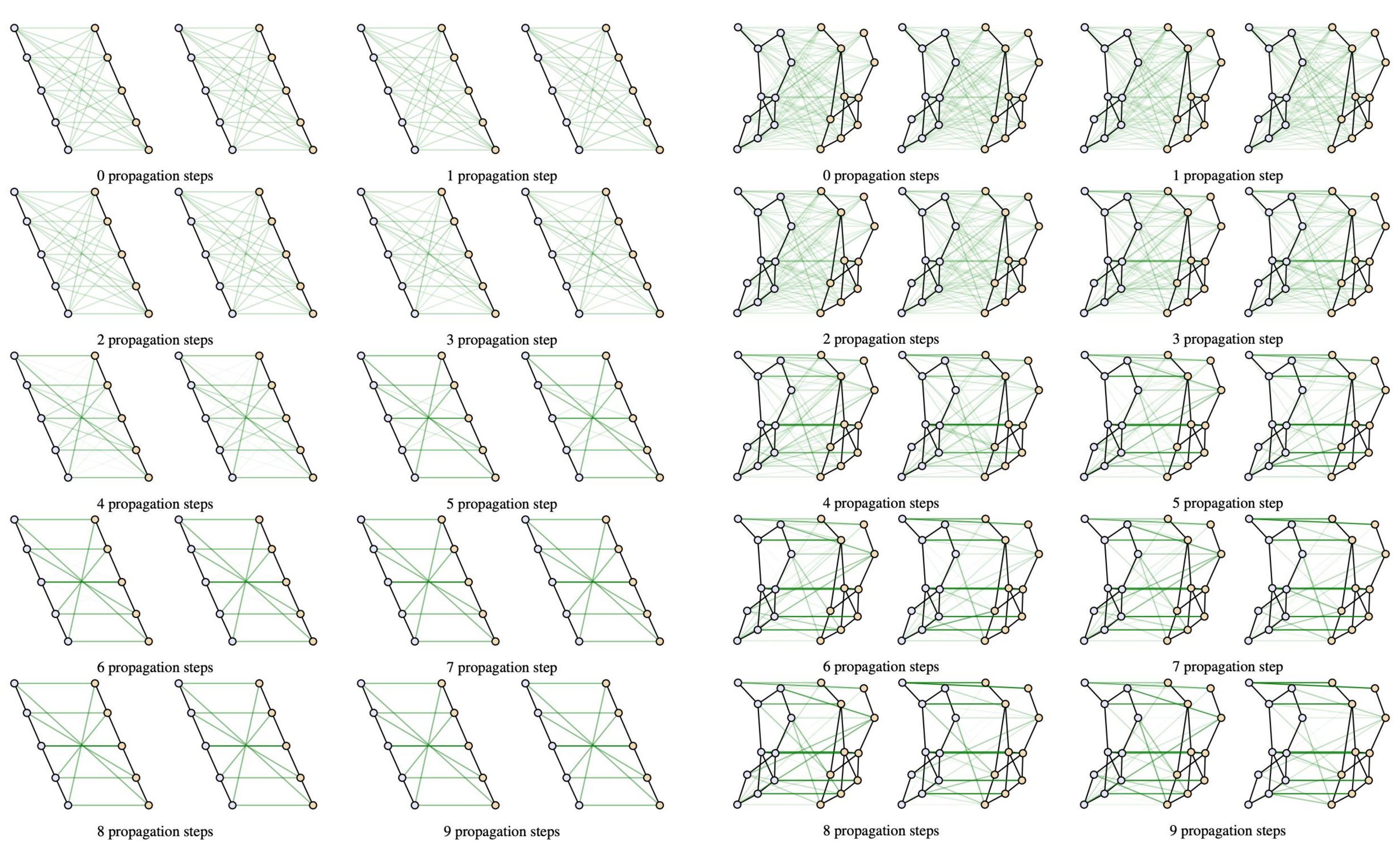
Triplet training:

$$L_{\mathrm{triplet}} = \mathbb{E}_{(G_1,G_2,G_3)}[\max\{0, d(G_1,G_2) - d(G_1,G_3) + \gamma\}]$$

(G_1,G_2) is a similar pair, (G_1,G_3) is a dissimilar pair, γ : margin

Attention Visualizations

We never supervise the cross-graph attention, but the model still learns some interesting attention patterns.



Conclusions, Limitations and Future Work

Graph similarity can be learned with graph neural networks.
Graph Matching Networks perform better than embedding models.

GMN is more expensive compared to GNN embedding models, requiring O(|V_1||V_2|) computation at each step.
- this provides us with an accuracy-computation trade-off

GMNs may be used jointly with GNNs embedding models in a retrieval system: GNN for fast filtering, GMN for refinement.

Future directions:
- larger graphs
- more effective / scalable attention
- different matching architectures
- many more!

arXiv paper