GraphGAN: Graph Representation Learning with Generative Adversarial Nets

Hongwei Wang\textsuperscript{1,2}, Jia Wang\textsuperscript{3}, Jialin Wang\textsuperscript{4}, Miao Zhao\textsuperscript{3}, Weinan Zhang\textsuperscript{1}, Fuzheng Zhang\textsuperscript{2}, Xing Xie\textsuperscript{2}, Minyi Guo\textsuperscript{1}

\textsuperscript{1} Shanghai Jiao Tong University, \textsuperscript{2} Microsoft Research Asia, \textsuperscript{3} The Hong Kong Polytechnic University, \textsuperscript{4} Huazhong University of Science and Technology

January 10, 2018
About Me

- **Hongwei Wang (王鸿伟)**
  - Ph.D. candidate at Shanghai Jiao Tong University
  - Full-time intern at Microsoft Research Asia
- **Advisor**
  - Prof. Minyi Guo (SJTU)
  - Dr. Xing Xie and Dr. Baining Guo (MSRA)
- **Research interests**
  - Recommender systems [TCSS 2017] [RecSys 2017] [CIKM 2017] [WWW 2018]
  - Graph representation learning [WSDM 2018] [AAAI 2018a]
  - Machine learning applications [ICDCS 2017] [TPDS 2018] [AAAI 2018b]
- **Homepage:** https://hwwang55.github.io
Outline

- Introduction to graph representation learning
  - Definition and application
  - Taxonomy
  - Representative work
- GraphGAN [AAAI 2018]
- GRL applications
  - Recommender systems [WWW 2018]
  - Sentiment prediction [WSDM 2018]

2. https://github.com/thunlp/NRLPapers
Outline

- **Introduction to graph representation learning**
  - Definition and application
  - Taxonomy
  - Representative work
- **GraphGAN** [AAAI 2018]
- **GRL application**
  - Recommender systems [WWW 2018]
  - Sentiment prediction [WSDM 2018]
Graph representation learning tries to embed each node of a graph into a low-dimensional vector space, which preserves the structural similarities or distances among the nodes in the original graph.

Also known as network embedding / graph embedding / network representation learning.
Graph representation learning can benefit a wide range of real-world applications:

- Link prediction (Gao, Denoyer, and Gallinari, CIKM 2011)
- Node classification (Tang, Aggarwal, and Liu, SDM 2016)
- Recommendation (Yu et al., WSDM 2014)
- Visualization (Maaten and Hinton, JMLR 2008)
- Knowledge graph representation (Lin et al., AAAI 2015)
- Clustering (Tian et al., AAAI 2014)
- Text embedding (Tang, Qu, and Mei, KDD 2015)
- Social network analysis (Liu et al., IJCAI 2016)
Taxonomy (1/3)

**Input**

- Homogeneous graph (e.g., citation network)
  - Weighted / Unweighted
  - Directed / Undirected
  - Signed / Unsigned

- Heterogeneous graph
  - Multimedia network
  - Knowledge graph

- Graph with side information
  - Node/edge label (categorical)
  - Node/edge attribute (discrete or continuous)
  - Node feature (e.g., texts)

- Graph transformed from non-relational data
  - Manifold learning
Taxonomy (2/3)

Output

- Node embedding (the most common case)
- Edge embedding
  - Relations in knowledge graph
  - Link prediction
- Sub-graph embedding
  - Substructure embedding
  - Community embedding
- Whole-graph embedding
  - Multiple small graphs, e.g., molecule, protein
Taxonomy (3/3)

Method

- Matrix factorization
  - Singular value decomposition
  - Spectral decomposition (eigen-decomposition)
- Random walk
- Deep learning
  - Auto-encoder
  - Convolutional neural network
- Self-defined loss
  - Maximizing edge reconstruction probability
  - Minimizing distance-based loss
  - Minimizing margin-based ranking loss
Representative Work

Long long ago
- PCA
- LDA
- MDS

Not long ago
- Isomap
  - Science, 2000
- LLE
  - Science, 2000
- LE
  - NIPS, 2001

Recent 5 years
- GraRep
  - CIKM, 2015
- HOPE
  - KDD, 2016
- LANE
  - WSDM, 2017
- Word2vec
  - NIPS, 2013
- DeepWalk
  - KDD, 2014
- Node2vec
  - KDD, 2016
- GraphEncoder
  - AAAI, 2014
- LINE
  - WWW, 2015
- APP
  - AAAI, 2017
- HNE
  - KDD, 2015
- SDNE
  - KDD, 2016
- HOPE
  - KDD, 2016
- TransR
  - AAAI, 2015
Linear Discriminant Analysis

- Suppose binary classification
- \( D = \{(x_i, y_i)\}, \) \( \mu_i \): mean of data of the \( i \)-th class, \( \Sigma_i \): covariance matrix of data of the \( i \)-th class
- Make projected covariance matrix as small as possible, while make projected distance between the mean of two classes as large as possible
- maximize

\[
J = \frac{\|w^T\mu_0 - w^T\mu_1\|^2}{w^T\Sigma_0 w + w^T\Sigma_1 w} = \frac{w^T(\mu_0 - \mu_1)(\mu_0 - \mu_1)^Tw}{w^T(\Sigma_0 + \Sigma_1)w}
\]

《机器学习》，周志华
Locally Linear Embedding

Introduction

- An unsupervised learning algorithm that computes low-dimensional, neighborhood-preserving embeddings of high-dimensional inputs
- LLE keeps linear dependency between local instances
Word2vec

- **Skip-Gram Model**
  - Find word representations that are useful for predicting the surrounding words in a sentence
  - Maximizing

\[
\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t)
\]

where

\[
p(w_o | w_I) = \frac{\exp(v'_{w_o} v_{w_I})}{\sum_{w=1}^{W} \exp(v'_{w} v_{w_I})}
\]

- **Negative sampling:**

\[
\log \sigma(v'_{w_o} v_{w_I}) + \sum_{k} \mathbb{E}_{w_i \sim P_n(w)} \left[ \log \sigma(-v'_{w_i} v_{w_I}) \right]
\]
DeepWalk / Node2vec

- **DeepWalk: Random walk + Word2vec**
  - Sample the next node to be visited **uniformly** from the neighbors of current node
  
  \[
  \minimize_{\Phi} \quad - \log \Pr(\{v_{i-w}, \ldots, v_{i+w}\} \setminus v_i | \Phi(v_i))
  \]
  - Hierarchical softmax

- **Node2vec: Biased random walk + Word2vec**
  - Sample the next node to be visited **with bias** from the neighbors of current node

![Diagram of BFS and DFS](image-url)
First-order proximity

\[ O_1 = d(\hat{p}_1(\cdot, \cdot), p_1(\cdot, \cdot)) \]
\[ p_1(v_i, v_j) = \frac{1}{1 + \exp(-\mathbf{u}_i^T \cdot \mathbf{u}_j)} \]
\[ \hat{p}_1(i, j) = \frac{w_{i,j}}{W} \]
\[ O_1 = - \sum_{(i, j) \in E} w_{i,j} \log p_1(v_i, v_j) \]

Second-order proximity

\[ O_2 = \sum_{i \in V} \lambda_i d(\hat{p}_2(\cdot|v_i), p_2(\cdot|v_i)) \]
\[ \hat{p}_2(v_j|v_i) = \frac{w_{i,j}}{d_i} \]
\[ p_2(v_j|v_i) = \frac{\exp(\mathbf{u}^T_j \cdot \mathbf{u}_i)}{\sum_{k=1}^{\vert V \vert} \exp(\mathbf{u}^T_k \cdot \mathbf{u}_i)} \]
\[ O_2 = - \sum_{(i,j) \in E} w_{i,j} \log p_2(v_j|v_i) \]
TransX

- Embed **knowledge graph** into a continuous vector space while preserving structural information.

- **TransE (NIPS 13):**
  - Ensures \( h + r \approx t \) when \((h, r, t)\) holds.

- **TransH (AAAI 14):**
  - Ensures \( h_\perp + r \approx t_\perp \) when \((h, r, t)\) holds, where \( h_\perp = h - w_r^T h w_r \) and \( t_\perp = t - w_r^T t w_r \).

- **TransR (AAAI 15):**
  - Score function: \( f_r(h, t) = \| h M_r + r - t M_r \|_2^2 \), where \( M_r \) is the projection matrix for relation \( r \).
SDNE

- **Reconstruction loss term**
  \[ \mathcal{L}_{2nd} = \sum_{i=1}^{n} \| (\hat{x}_i - x_i) \odot b_1 \|_2^2 \]
  \[ = \| (\hat{X} - X) \odot B \|_F^2 \]

- **Proximity loss term**
  \[ \mathcal{L}_{1st} = \sum_{i,j=1}^{n} s_{i,j} \| y_i^{(K)} - y_j^{(K)} \|_2^2 \]
  \[ = \sum_{i,j=1}^{n} s_{i,j} \| y_i - y_j \|_2^2 \]

- **Regularization term**
  \[ \mathcal{L}_{reg} = \frac{1}{2} \sum_{k=1}^{K} (\| W^{(k)} \|_F^2 + \| \tilde{W}^{(k)} \|_F^2) \]

- **Loss function**
  \[ \mathcal{L}_{mix} = \mathcal{L}_{2nd} + \alpha \mathcal{L}_{1st} + \nu \mathcal{L}_{reg} \]
Outline

- Introduction to graph representation learning
  - Definition and application
  - Taxonomy
  - Representative work
- **GraphGAN [AAAI 2018]**
- **GRL application**
  - Recommender systems [WWW 2018]
  - Sentiment prediction [WSDM 2018]
Motivation (1/3)

Generative Model

- Generative graph representation learning model assumes an **underlying true connectivity distribution** $p_{true}(v|v_c)$ for each vertex $v_c$
  - The edges can be viewed as observed samples generated by $p_{true}(v|v_c)$
  - Vertex embeddings are learned by maximizing the likelihood of edges
  - E.g., DeepWalk (KDD 2014) and node2vec (KDD 2016)
Motivation (2/3)

Discriminative Model

- Discriminative graph representation learning model aim to learn a **classifier** for predicting the existence of edges directly
  - Consider two vertices $v_i$ and $v_j$ jointly as features
  - Predict the probability of an edge existing between them, i.e., $p(edge|v_i, v_j)$
  - E.g., SDNE (KDD 2016) and PPNE (DASFAA, 2017)

$$p(edge|v_i, v_j) = 0.8$$
$$p(edge|v_i, v_k) = 0.3$$

......
Motivation (3/3)

G + D ?

- Generative and discriminative models are two sides of the same coin
- LINE (WWW 2015) has tried to combine these two objectives
- Generative adversarial nets (GAN) have received a great deal of attention
  - GAN designs a game-theoretical minimax game to combine G and D
  - GAN achieves success in various applications:
    - image generation (Denton et al., NIPS 2015)
    - sequence generation (Yu et al., AAAI 2017)
    - dialogue generation (Li et al., arXiv 2017)
    - information retrieval (Wang et al., SIGIR 2017)
    - domain adaption (Zhang, Barzilay, and Jaakkola, arXiv 2017)

- We propose GraphGAN, a framework that unifies generative and discriminative thinking for graph representation learning
The Minimax Game

- \( G = (\mathcal{V}, \mathcal{E}) \), \( \mathcal{V} = \{v_1, \ldots, v_V\} \), \( \mathcal{E} = \{e_{ij}\}_{i,j=1}^V \)

- \( \mathcal{N}(v_c) \): set of neighbors of \( v_c \)

- \( p_{true}(v_c) \): underlying true connectivity distribution for \( v_c \)

- The objective of GraphGAN is to learn the following two models:
  - \( G(v|v_c; \theta_G) \) which tries to approximate \( p_{true}(v_c) \)
  - \( D(v, v_c; \theta_D) \) which aims to discriminate the connectivity for the vertex pair \((v, v_c)\)

- The two-player minimax game:

\[
\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^{V} \left( \mathbb{E}_{v \sim p_{true}(\cdot|v_c)} \left[ \log D(v, v_c; \theta_D) \right] + \mathbb{E}_{v \sim G(\cdot|v_c; \theta_G)} \left[ \log \left( 1 - D(v, v_c; \theta_D) \right) \right] \right)
\]
Implementation & Optimization of D

\[
\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^{V} \left( \mathbb{E}_{v \sim p_{\text{true}}(\cdot|v_c)} \left[ \log D(v, v_c; \theta_D) \right] + \mathbb{E}_{v \sim G(\cdot|v_c; \theta_G)} \left[ \log \left( 1 - D(v, v_c; \theta_D) \right) \right] \right)
\]  

(1)

- **Implementation of D:**

  \[D(v, v_c; \theta_D) = \sigma(\mathbf{d}_v^T \mathbf{d}_{v_c}) = \frac{1}{1 + \exp(-\mathbf{d}_v^T \mathbf{d}_{v_c})},\]  

  (2)

  where \(\mathbf{d}_v, \mathbf{d}_{v_c} \in \mathbb{R}^k\) are the \(k\)-dimensional vectors of \(v\) and \(v_c\) for D

- **Gradient of** \(V(G, D)\) **w.r.t** \(\theta_D\):

  \[\nabla_{\theta_D} V(G, D) = \begin{cases} 
\nabla_{\mathbf{d}_v, \mathbf{d}_{v_c}} \log D(v, v_c; \theta_D), & \text{if } v \sim p_{\text{true}}; \\
\nabla_{\mathbf{d}_v, \mathbf{d}_{v_c}} \left(1 - \log D(v, v_c; \theta_D)\right), & \text{if } v \sim G.
\end{cases}\]  

(3)
Optimization of $G$

$$
\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^{V} \left( \mathbb{E}_{v \sim p_{true}(\cdot | v_c)} \left[ \log D(v, v_c; \theta_D) \right] + \mathbb{E}_{v \sim G(\cdot | v_c; \theta_G)} \left[ \log (1 - D(v, v_c; \theta_D)) \right] \right)
$$

(1)

- Gradient of $V(G, D)$ w.r.t $\theta_G$ (policy gradient):

$$
\nabla_{\theta_G} V(G, D) = \nabla_{\theta_G} \sum_{c=1}^{V} \mathbb{E}_{v \sim G(\cdot | v_c; \theta_G)} \left[ \log (1 - D(v, v_c; \theta_D)) \right]
$$

$$
= \sum_{c=1}^{V} \sum_{i=1}^{N} G(v_i | v_c; \theta_G) \nabla_{\theta_G} \log G(v_i | v_c; \theta_G) \log (1 - D(v_i, v_c; \theta_D))
$$

$$
= \sum_{c=1}^{V} \sum_{i=1}^{N} G(v_i | v_c; \theta_G) \nabla_{\theta_G} \log G(v_i | v_c; \theta_G) \log (1 - D(v_i, v_c; \theta_D))
$$

$$
= \sum_{c=1}^{V} \mathbb{E}_{v \sim G(\cdot | v_c; \theta_G)} \left[ \nabla_{\theta_G} \log G(v | v_c; \theta_G) \log (1 - D(v, v_c; \theta_D)) \right].
$$

(4)
GraphGAN Framework

\[
\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^{V} \left( \mathbb{E}_{v \sim p_{true}(\cdot|v_c)} \left[ \log D(v, v_c; \theta_D) \right] + \mathbb{E}_{v \sim G(\cdot|v_c; \theta_G)} \left[ \log \left( 1 - D(v, v_c; \theta_D) \right) \right] \right)
\]

(1)
Implementation of G

- Softmax?
  - Computationally inefficient
  - Graph-structure-unaware

- Hierarchical softmax?
  - Graph-structure-unaware

- Negative sampling?
  - Not a valid probability distribution
  - graph-structure-unaware

\[
G(v|v_c; \theta_G) = \frac{\exp(g_v^T g_{vc})}{\sum_{v\neq v_c} \exp(g_v^T g_{vc})}
\]

where \(g_v, g_{vc} \in \mathbb{R}^k\) are the \(k\)-dimensional vectors of \(v\) and \(v_c\) for \(G\)

\[
p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma \left( \left[ n(w, j + 1) = \text{ch}(n(w, j)) \right] \cdot v'_{n(w,j)}^T v_{w_I} \right)
\]

\[
\log \sigma(v'_{wO}^T v_{wI}) + \sum_{i=1}^{k} E_{w_i \sim P_n(w)} \left[ \log \sigma(-v'_{w_i}^T v_{wI}) \right]
\]
Graph Softmax (1/5)

Objectives

- The design of graph softmax should satisfy the following three properties:

  - **Normalized**: The generator should produce a valid probability distribution, i.e.,
    \[ \sum_{v \neq v_c} G(v|v_c; \theta_G) = 1 \]

  - **Graph-structure-aware**: The generator should take advantage of the structural information of a graph

  - **Computationally efficient**: The computation of \( G(v|v_c; \theta_G) \) should only involve a small number of vertices in the graph
Graph Softmax (2/5)

Design

- **Breadth First Search** (BFS) on $G$ from every vertex $v_c$
  - **BFS-tree** $T_c$ rooted at $v_c$
- For a given vertex $v$ and one of its neighbors $v_i \in \mathcal{N}_c(v)$, the **relevance probability** of $v_i$ given $v$ as
  \[
  p_c(v_i|v) = \frac{\exp(g_{v_i}^T g_v)}{\sum_{v_j \in \mathcal{N}_c(v)} \exp(g_{v_j}^T g_v)},
  \]
  where $\mathcal{N}_c(v)$ is the set of neighbors of $v$ in $T_c$
- **Graph softmax**
  \[
  G(v|v_c; \theta_G) \triangleq \left( \prod_{j=1}^{m} p_c(v_{r_j}|v_{r_{j-1}}) \right) \cdot p_c(v_{r_m}|v_{r_m}),
  \]
  given the unique path from $v_c$ to $v$ in tree $T_c$: $P_{v_c \to v} = (v_{r_0}, v_1, ..., v_{r_m})$, where $v_{r_0} = v_c$ and $v_{r_0} = v$
Design

- **Graph softmax**

\[
G(v|v_c; \theta_G) \triangleq \left( \prod_{j=1}^{m} p_c(v_{r_j} | v_{r_{j-1}}) \right) \cdot p_c(v_{r_{m-1}} | v_{r_m}),
\]

given the unique path from \(v_c\) to \(v\) in tree \(T_c\): \(P_{v_c \rightarrow v} = (v_{r_0}, v_1, ..., v_{r_m})\), where \(v_{r_0} = v_c\) and \(v_{r_0} = v\)
Graph Softmax (4/5)

Properties

- $\sum_{v \neq v_c} G(v|v_c; \theta_G) = 1$ in graph softmax

- In graph softmax, $G(v|v_c; \theta_G)$ decreases exponentially with the increase of the shortest distance between $v$ and $v_c$ in original graph $G$

- In graph softmax, calculation of $G(v|v_c; \theta_G)$ depends on $O(d \log V)$ vertices, where $d$ is average degree of vertices and $V$ is the number of vertices in graph $G$
Graph Softmax (5/5)

Generating Strategy

Algorithm 1 Online generating strategy for the generator

Input: BFS-tree $T_c$, representation vectors $\{g_i\}_{i \in V}$
Output: generated sample $v_{gen}$

1: $v_{pre} \leftarrow v_c, v_{cur} \leftarrow v_c$
2: while true do
3: Randomly select $v_i$ proportionally to $p_c(v_i|v_{cur})$ in Eq. (6);
4: if $v_i = v_{pre}$ then
5: $v_{gen} \leftarrow v_{cur}$;
6: return $v_{gen}$
7: else
8: $v_{pre} \leftarrow v_{cur}, v_{cur} \leftarrow v_i$;
9: end if
10: end while
Algorithm 2 GraphGAN framework

**Input:** dimension of embedding $k$, size of generating samples $s$, size of discriminating samples $t$

**Output:** generator $G(v|v_c; \theta_G)$, discriminator $D(v, v_c; \theta_D)$

1. Initialize and pre-train $G(v|v_c; \theta_G)$ and $D(v, v_c; \theta_D)$;
2. Construct BFS-tree $T_c$ for all $v_c \in \mathcal{V}$;
3. while GraphGAN not converge do
   4. for $G$-steps do
      5. $G(v|v_c; \theta_G)$ generates $s$ vertices for each vertex $v_c$ according to Algorithm 1;
      6. Update $\theta_G$ according to Eq. (4), (6) and (7);
   7. end for
   8. for $D$-steps do
      9. Sample $t$ positive vertices from ground truth and $t$ negative vertices from $G(v|v_c; \theta_G)$ for each vertex $v_c$;
   10. Update $\theta_D$ according to Eq. (2) and (3);
   11. end for
   12. end while
13. return $G(v|v_c; \theta_G)$ and $D(v, v_c; \theta_D)$
Experiments (1/3)

Datasets
- arXiv-AstroPh: 18,772 vertices and 198,110 edges
- arXiv-GrQc: 5,242 vertices and 14,496 edges
- BlogCatalog: 10,312 vertices, 333,982 edges and 39 labels
- Wikipedia: 4,777 vertices, 184,812 edges and 40 labels
- MovieLens-1M: 6,040 users and 3,706 movies

Baselines
- DeepWalk (KDD 2014)
- LINE (WWW 2015)
- Node2vec (KDD 2016)
- Struc2vec (KDD 2017)
Experiments (2/3)

Link Prediction

- Learning curves

![Learning Curves](image)

Fig. 4: Learning curves of the generator and the discriminator of GraphGAN on arXiv-GrQc in link prediction.

- Results

<table>
<thead>
<tr>
<th>Model</th>
<th>arXiv-AstroPh</th>
<th>arXiv-GrQc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Macro-F1</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>0.841</td>
<td>0.839</td>
</tr>
<tr>
<td>LINE</td>
<td>0.820</td>
<td>0.814</td>
</tr>
<tr>
<td>Node2vec</td>
<td>0.845</td>
<td>0.854</td>
</tr>
<tr>
<td>Struc2vec</td>
<td>0.821</td>
<td>0.810</td>
</tr>
<tr>
<td>GraphGAN</td>
<td><strong>0.855</strong></td>
<td><strong>0.859</strong></td>
</tr>
</tbody>
</table>
Experiments (3/3)

Node Classification

TABLE 2: Accuracy and Macro-F1 on BlogCatalog and Wikipedia in node classification.

<table>
<thead>
<tr>
<th>Model</th>
<th>BlogCatalog</th>
<th>Wikipedia</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Macro-F1</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>0.225</td>
<td>0.214</td>
</tr>
<tr>
<td>LINE</td>
<td>0.205</td>
<td>0.192</td>
</tr>
<tr>
<td>Node2vec</td>
<td>0.215</td>
<td>0.206</td>
</tr>
<tr>
<td>Struc2vec</td>
<td>0.228</td>
<td>0.216</td>
</tr>
<tr>
<td>GraphGAN</td>
<td>0.232</td>
<td>0.221</td>
</tr>
</tbody>
</table>

Recommendation

Fig. 5: Precision@K and Recall@K on MovieLens-1M in recommendation.
We propose **GraphGAN**, a novel framework that unifies generative and discriminative thinking for graph representation learning

- Generator $G(v|v_c)$ tries to fit $p_{true}(v|v_c)$ as much as possible
- Discriminator $D(v, v_c)$ tries to tell whether an edge exists between $v$ and $v_c$

G and D act as two players in a **minimax game**:

- G tries to produce the most indistinguishable “fake” vertices under guidance provided by D
- D tries to draw a clear line between the ground truth and “counterfeits” to avoid being fooled by G

We propose **graph softmax** as the implementation of G

- Graph softmax overcomes the limitations of softmax and hierarchical softmax
- Graph softmax satisfies the properties of **normalization**, **graph structure awareness** and **computational efficiency**
Outline

- Introduction to graph representation learning
  - Definition and application
  - Taxonomy
  - Representative work
- GraphGAN [AAAI 2018]
- GRL application
  - Recommender systems [WWW 2018]
  - Sentiment prediction [WSDM 2018]
DKN (WWW 2018)

- **DKN: Deep Knowledge-Aware Network for News Recommendation**
  - Learning knowledge graph representations by TransX
  - A CNN framework for combining word embedding and entity embedding
  - Attention-based CTR prediction

Fig. 1: Knowledge graph in news recommendation

Fig. 2: DKN framework
SHINE: Signed Heterogeneous Information Network Embedding for Sentiment Link Prediction

- **Sign**: positive/negative sentiment link
- **Heterogeneity**: sentiment network, social network, knowledge graph
- **Auto-encoder based framework**

Fig. 1: Signed heterogeneous networks in sentiment prediction

Fig. 2: Auto-encoder for sentiment network representation learning
SHINE: Signed Heterogeneous Information Network Embedding for Sentiment Link Prediction

- **Sign**: positive/negative sentiment link
- **Heterogeneity**: sentiment network, social network, knowledge graph
- **Auto-encoder based framework**

Fig. 3: SHINE framework
Q & A

Thanks!

Visit https://hwwang55.github.io for full papers, slides, code and more information