Graph Embeddings for Natural Language Processing

Lecture at ESSLLI 2022

DOI: 10.5281/zenodo.6667766

Outline

Section 1

Introduction

Introduction

- Linguistic data are sparse, so the graphs are usually sparse, too
- Modern Natural Language Processing (NLP) is based on embeddings and representation learning
- We would like to reduce the dimensionality, but keep the important graph properties

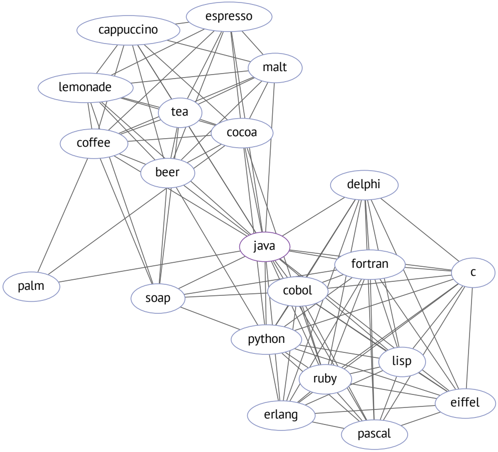
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Core Idea: Embed the Graphs Wisely

We can incorporate the relationships between objects in our machine learning pipelines.

Motivation

Remember this *distributional thesaurus*?



Source: Ustalov et al. (2019)

- Can we measure the similarity between "tea" and "lisp"?
- Can we employ the node relationships as features?
- **Yes.**

Successful Applications

Graph embeddings help in addressing very challenging NLP problems:

- question answering (Bordes et al., 2014)
- semantic role labeling (Marcheggiani et al., 2017)
- text-to-entity mapping (Kartsaklis et al., 2018)
- text classification (Yao et al., 2019)
- fact-checking (Zhong et al., 2020)
- explanation regeneration (Li et al., 2020)

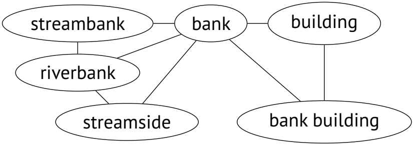
Beyond these applications, graph embeddings are generally useful for

- node classification, recommendation, and link prediction
- feature extraction
- visualization (not every approach performs a proper layout)

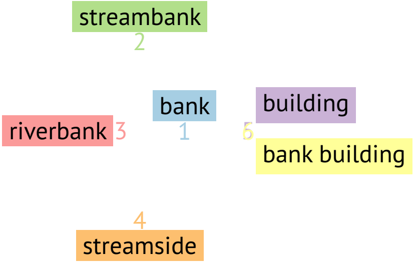
Problem Formulation

- There are node embeddings, edge embeddings, and the whole graph embeddings; we will focus on *node embeddings*
- Given a graph $G = (V, E)$ and a number of dimensions $d \ll |V|$, we map G into a d-dimensional space, in which the certain *graph property* is preserved as much as possible (Cai et al., 2018)
- Usually, we would like to minimize some loss function using gradient-based optimization (Goodfellow et al., 2016)

Input Graph



Output Embedding



Section 2

Unsupervised Embeddings

Unsupervised Embeddings

- Unsupervised node embeddings build representations preserving generic graph properties
- We will focus on two different graph embedding methods: Laplacian Eigenmaps and DeepWalk
- There are *a lot* of other methods, see Cai et al. (2018) and Goyal et al. (2018)

Source: Finnsson (2017)

Laplacian Eigenmaps (Spectral Embeddings)

- **Laplacian Eigenmaps** is a spectral approach for embedding high-dimensional data (Belkin et al., 2003)
- Compute a normalized Laplacian of the graph and run (approximate) *eigenvalue decomposition* to obtain the node embeddings; similar to spectral clustering (Ng et al., 2002)
- Preserved graph properties are pairwise node similarities

Source: Amos (2011)

Laplacian Eigenmaps: Algorithm

Input: graph $G = (V, E)$, adjacency matrix A, degree matrix D, dimensions $d \ll |V|$

Output: embedding $\vec{u} \in \mathbb{R}^d, \forall u \in V$

1: $L^{\text{norm}} \leftarrow D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$

2: $U\Lambda U^{-1} \leftarrow \texttt{ein}(L^{\text{norm}})$ ▷ Assume the eigenvalues are descending

3: $U' \leftarrow (U_{ik})_{\substack{1 \leq i \leq |V|, 1 \leq j \leq d \\ k = |V| - 1 - j}}$ ▷ Drop the smallest eigenvalue

4: **return** $\vec{u}_i \rightarrow U'_i$ **for all** $1 \leq i \leq |V|$

Laplacian Eigenmaps: Example

$$U' = \begin{pmatrix} .06 & 0 \\ -.31 & .71 \\ -.45 & 0 \\ -.31 & -.71 \\ .55 & 0 \\ 55 & 0 \end{pmatrix}$$

- This is an example using the graph from Ustalov et al. (2019, Figure 2)

Laplacian Eigenmaps: Discussion

Pros:

- + Sound method that preserves local information optimally
- + Very simple to implement

Cons:

- − Slow, the worst-case running time is $O(|E|d^2)$
- − Preserves only first-order proximity
- − Graph should have only one connected component

Implementation:

- https://scikit-learn.org/stable/modules/generated/sklearn.manifold.spectral_embedding.html

Word2Vec Recap

- Mikolov et al. (2013) proposed Word2Vec, an efficient technique for learning *distributional representations* of words
- For each pair of word w and its context c in the fixed window, the Skip-Gram method performs negative sampling of $k \in \mathbb{N}$ contexts from a distribution P_D and computes the objective (Levy et al., 2014):

$$\log \sigma(\vec{w} \cdot \vec{c}) + k \cdot \mathbb{E}_{c_N \sim P_D} \left[\log \sigma(-\vec{w} \cdot \vec{c}_N)\right]$$

- Example representations:
  $\overrightarrow{\text{Paris}} - \overrightarrow{\text{France}} + \overrightarrow{\text{Russia}} \approx \overrightarrow{\text{Moscow}}$
  $\overrightarrow{\text{apple}} - \overrightarrow{\text{apples}} \approx \overrightarrow{\text{car}} - \overrightarrow{\text{cars}}$
- Popular variations are CBOW (Mikolov et al., 2013; İrsoy et al., 2021), GloVe (Pennington et al., 2014), *fast*Text (Bojanowski et al., 2017), etc.

DeepWalk

- **DeepWalk** uses truncated random walks to learn latent representations by treating walks as the equivalent of *natural language sentences* (Perozzi et al., 2014)
- The input graph is flattened into a "corpus" of fixed-size node sequences; this corpus is used to train a Word2Vec model (Mikolov et al., 2013)

Source: Pexels (2016)

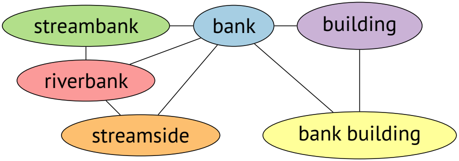
DeepWalk: Algorithm

Input: graph $G = (V, E)$, dimensions $d \ll |V|$, window size $w \in \mathbb{N}$,
walks per node $\gamma \in \mathbb{N}$, walk length $t \in \mathbb{N}$, learning rate $\alpha > 0$

Output: embedding $\vec{u} \in \mathbb{R}^d, \forall u \in V$

$\Phi \leftarrow \texttt{random}(\mathbb{R}^{|V| \times d})$ ▷ Initialize from a uniform distribution
for $i \leftarrow 0 \ldots \gamma$ **do**
    for all $u \in V$ in random order **do**
        $\mathcal{W}_u \leftarrow \texttt{walk}(G, u, t)$ ▷ Random walk of length t from u
        $\Phi \leftarrow \texttt{Skip-Gram}(\mathcal{W}_u, w, \alpha, \Phi)$ ▷ Update the parameters
return $\vec{u}_i \rightarrow \Phi_i$ **for all** $1 \leq i \leq |V|$

DeepWalk: Example



This is an example using the graph from Ustalov et al. (2019, Figure 2)

DeepWalk: Discussion

Pros:

- + Very simple and works very well in practice
- + Fast, the number of parameters is $O(d|V|)$

Cons:

- − Does not preserve community structure
- − Does not preserve structural equivalence between nodes
- − Edge weights are ignored (more on this a bit later)

Implementation:

- https://github.com/phanein/deepwalk
- https://snap.stanford.edu/node2vec/
- http://rdf2vec.org/

Word2Vec as Implicit Matrix Factorization

Levy et al. (2014) showed that Skip-Gram is an implicit factorization of a pointwise mutual information (PMI) word-context matrix.

- Given the word $w \in V$ and its context c, we count the number of words in context:

$$\mathrm{PMI}(w, c) = \log \frac{\#(w, c) \cdot |D|}{\#(w) \cdot \#(c)}$$

- We obtain a *shifted PMI* matrix by shifting the PMI by a constant offset:

$$\mathrm{SPPMI}_k(w, c) = \max(\mathrm{PMI}(w, c) - \log k, 0)$$

- A truncated singular value decomposition $M^{\mathrm{SPPMI}_k} = U_d \Sigma_d V_d^\top$ for the rank d (Hansen, 1987) allows obtaining the embeddings $\Phi = U_d \sqrt{\Sigma_d}$ (here V_d is a matrix and not a subset of V)

“Embed All the Things!”

Wu et al. (2018) proposed a general-purpose embedding model **StarSpace**:

$$\sum_{(a,b)\in E^+} \sum_{b^-\in E^-} \underbrace{\max(0, \mu - \mathrm{sim}(a,b) + \mathrm{sim}(a,b^-))}_{\text{margin ranking loss, } \mu \in \mathbb{R}}$$

- Positive pairs E^+ are task-dependent and provided as the input
- Negative pairs E^- are obtained by choosing $k \in \mathbb{N}$ negative pairs randomly
- Similarity function sim is either a dot product or cosine
- StarSpace is a convenient strong baseline for many tasks involving embedding entities comprised of discrete features: https://github.com/facebookresearch/StarSpace

Unsupervised Embeddings: Wrap-Up

- Unsupervised node embeddings capture meaningful representations that can be concatenated or fine-tuned for downstream applications
- Edge weights can be handled by performing graph traversal with BFS and DFS (Grover et al., 2016) or biased walks (Kartsaklis et al., 2018; Ristoski et al., 2018)
- Textual features of graph nodes can be incorporated into embeddings (Yang et al., 2015)

Source: rawpixel (2017)

Section 3

Graph Neural Networks

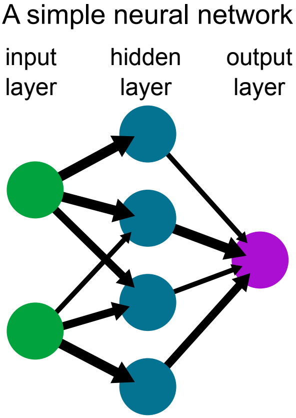
Graph Neural Networks

- Building embeddings is not the ultimate goal: they are used in applications and there are useful features of the nodes
- **Graph Neural Networks** (GNNs) use the node features and relationships to learn node or graph representations
- We will focus on the two most common GNN models, GCN and GAT, but there are many others, see Wu et al. (2022)

Source: McGuire (2015)

Neural Networks Recap

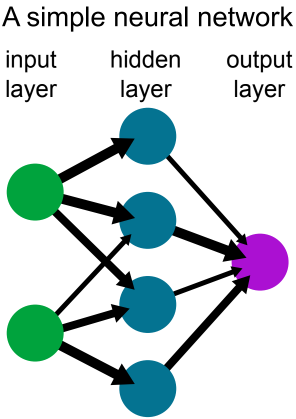
- We will consider a neural network (NN) as a sequence of non-linear transformations of the input data X called layers; the output of each layer is the input for the next one
- Parameters are estimated using backpropagation, see more in Goodfellow et al. (2016, Chapter 6)
- For convenience, we will omit the bias terms, so the output of each layer is $H = \sigma(XW)$, where weights W are trainable parameters and σ is an activation function, such as $\tanh$, ReLU, etc.



Source: Wiso (2008)

Graph Neural Networks

- We can think of rows of $X \in \mathbb{R}^{|V| \times n}$ as graph nodes, and we can think of its columns as n-dimensional node features, $n \in \mathbb{N}$
- GNNs aim at including node relationships in the model; so given the number of embedding dimensions $d \in \mathbb{N}$, we will denote the next GNN layer as $H' \in \mathbb{R}^{|V| \times d}$
- We will denote the embedding of node $u \in V$ as $\vec{h'}_u \in \mathbb{R}^d$



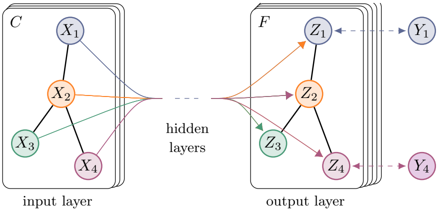
Source: Wiso (2008)

Graph Convolutional Network

Kipf et al. (2017) proposed a **Graph Convolutional Network** (GCN), a simple and theoretically-motivated layer-wise propagation rule for NNs.

- Instead of propagating $\sigma(XW)$, we insert information about node relationships $\hat{A}$, so the propagation rule becomes $H' = \sigma(\hat{A}HW)$
- To avoid numerical instability, we perform a *renormalization trick* with adjacency matrix: $\tilde{A} = A + I$ and $\tilde{D}_{ii} = \sum_{1 \leq j \leq |V|} \tilde{A}_{ij}$, so $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$

$$H' = Z = \hat{A}XW$$



Source: Kipf et al. (2017)

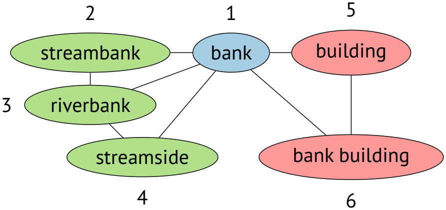
Graph Convolutional Network: Estimation

As a semi-supervised method, GCN relies on labeled nodes $V_L \subseteq V$, and can be trained using the cross-entropy loss:

$$-\sum_{u_l \in V_L} \sum_{f=1}^{F} Y_{lf} \log Z_{lf},$$

where $Y_{lf} = \begin{cases} 1, & \text{if } u_l \in V_L \text{ belongs to class } f, \\ 0, & \text{otherwise} \end{cases}$

- Note that $F \ll |V|$ is the target number of dimensions d



$$\begin{pmatrix} -.13 & -.87 \\ .89 & -.85 \\ .96 & -.85 \\ .90 & -.85 \\ -.98 & -.80 \\ -.98 & -.80 \end{pmatrix}$$

- This is an example using the graph from Ustalov et al. (2019, Figure 2)

Graph Convolutional Network: Discussion

Pros:

- ＋ Sound method that approximates localized spectral filters on graphs
- ＋ Fast, the running time is linear in the number of edges

Cons:

- − Prone to over-smoothing (Chen et al., 2020)
- − Exact algorithm requires the complete $\hat{A}$, but sampling could help (Hamilton et al., 2017)

Implementations:

- https://github.com/tkipf/gcn
- https://github.com/tkipf/pygcn

There are variations of GCN, such as TextGCN (Yao et al., 2019), that learn predictive word and document embeddings for text classification.

Graph Attention Network

Veličković et al. (2018) proposed **Graph Attention Network** (GAT) that leverages self-attention layers to learn neighbor importances α_{ij} for all $u_i \in V$ and $u_j \in V$.

GAT computes $K \in \mathbb{N}$ attentions per layer and then concatenates them:

$$\vec{h'}_i = \Big\|_{k=1}^{K} \sigma\left(\sum_{v_j \in V_{v_i}} \alpha_{ij}^{(k)} W^{(k)} \vec{h}_j\right)$$

At the final (prediction) layer, averaging is performed:

$$\vec{h'}_i = \sigma\left(\frac{1}{K} \sum_{k=1}^{K} \sum_{v_j \in V_v} \alpha_{ij}^{(k)} W^{(k)} \vec{h}_j\right)$$

Graph Attention Network: Attention

Self-attention on the nodes is parameterized by the vector $\vec{a} \in \mathbb{R}^{2d}$.

We compute attention coefficients $e_{ij} \in \mathbb{R}$ only for the adjacent nodes and use a modified definition of them (Brody et al., 2022):

$$e_{ij} = \mathrm{LeakyReLU}\left(\vec{a}^\top W[\vec{h_i} \parallel \vec{h_j}]\right)$$

To allow attention scores to be compared across different nodes, GAT computes the k-th normalized score $\alpha_{ij}^{(k)}$:

$$\alpha_{ij}^{(k)} = \frac{\exp(e_{ij})}{\sum_{u_l \in V_i} \exp(e_{il})}$$

Many implementations perform addition instead of concatenation in e_{ij}, so $\vec{a} \in \mathbb{R}^d$, and in our examples, we will use this configuration.

$\vec{a} = (.15, .84)^\top$

$$\alpha = \begin{pmatrix} 0 & .40 & .32 & .39 & .38 & .40 \\ .20 & 0 & .34 & 0 & 0 & 0 \\ .12 & .60 & 0 & .61 & 0 & 0 \\ .17 & 0 & .34 & 0 & 0 & 0 \\ .15 & 0 & 0 & 0 & 0 & .60 \\ .37 & 0 & 0 & 0 & .62 & 0 \end{pmatrix}$$

bank
building
riverbank
stream
streambank
bank building

$$\begin{pmatrix} -.05 & .07 \\ -.80 & -.68 \\ -.72 & -.41 \\ -.80 & -.67 \\ .49 & -.07 \\ .71 & -.68 \end{pmatrix}$$

This is an example using the graph from Ustalov et al. (2019, Figure 2)

Graph Attention Network: Discussion

Pros:

- Simple enough and work reasonably well in most benchmarks
- Estimates edge importances (α values)

Cons:

- Still prone to over-smoothing
- More sophisticated methods have been created since then

Implementations:

- https://github.com/PetarV-/GAT
- https://github.com/tech-srl/how_attentive_are_gats

A very well-written detailed annotated walkthrough is available at https://nn.labml.ai/graphs/gatv2/.

Graph Neural Networks: Wrap-Up

- Node embeddings can be efficiently estimated for the specific task
- These representations can be learned and extracted from the neural networks
- Semi-supervised representations do not require the complete data annotation
- Even a single layer of a GNN improves quality in practice (we will look at case studies)

Source: FreePhotosART (2016)

Section 4

Case Studies

Case Studies

- Embedding a Distributional Thesaurus (Jana et al., 2018)
- Mapping Text to Knowledge Graphs (Kartsaklis et al., 2018)
- Semantic Role Labeling (Marcheggiani et al., 2017)
- Explanation Regeneration (Jansen et al., 2020)

Source: Simone_ph (2017)

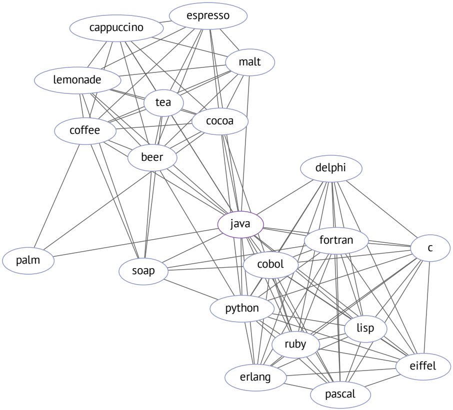
Embedding DTs

- Jana et al. (2018) used embeddings of nodes in a distributional thesaurus (DT) as additional features for building better word representations

Source: Buissinne (2016)

Embedding DTs: Approach

1. Build a distributional thesaurus (Biemann et al., 2013)
2. Learn node embeddings (DeepWalk, node2vec, etc.)
3. Concatenate node embeddings with GloVe word embeddings (Pennington et al., 2014)
4. Perform a principal component analysis (PCA)



Source: Ustalov et al. (2019)

Embedding DTs: Results

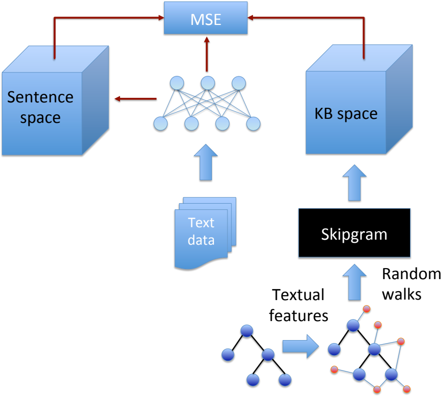
- According to Spearman's ρ, concatenation (CC) of GloVe vectors with the DeepWalk embeddings improved the results on multiple datasets
- Note that PCA also improved upon CC despite the loss of information after dimensionality reduction from $300 + 128$ to 300

Dataset	GloVe	CC	PCA
WSSim	0.799	0.838	0.839
SimL-N	0.427	0.443	0.468
RG-65	0.791	0.816	**0.879**
MC-30	0.799	0.860	**0.890**
WSR	0.637	**0.676**	0.645
M771	0.707	0.708	0.707
M287	0.800	0.781	0.807
MEN-N	**0.819**	0.792	0.799
WS-353	0.706	**0.751**	0.740

Source: Jana et al. (2018)

Text-to-Entity Mapping

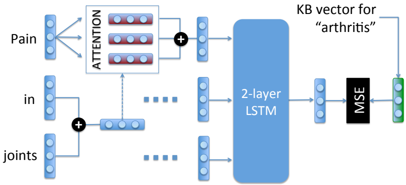
- Kartsaklis et al. (2018) proposed a technique for enriching the entity vectors with textual information
- Textual information is obtained from BabelNet (Navigli et al., 2012) and other sources



Source: Kartsaklis et al. (2018)

Text-to-Entity Mapping: Approach

1. Learn node embeddings with DeepWalk (Perozzi et al., 2014)
2. Build LSTM (Hochreiter et al., 1997) with multi-sense aspect (*aka* MS-LSTM)
3. Minimize the mean squared error (MSE) between the sense vector and the target entity vector



Source: Kartsaklis et al. (2018)

Code and Data: `https://bitbucket.org/dimkart/ms-lstm`

Text-to-Entity Mapping: Example

table^1	formulation, uncommonly, rauwolfia, cardiology, hypodermic, malleability, points, optic, dendrite, rubiaceae, nonparametric, meninges, deviation, anesthetics
table^2	tableware, meal, expectation, heartily, kitchen, hum, eating, forestay, suitors, croupier, companionship, restaurant, dishes, candles, cup, tea
table^3	reassigned, projective, ultracentrifuge, polemoniaceous, thyronine, assumptions, lymphocyte, atomic, difficulties, intracellular, virgil, elementary, cartesian

Source: Kartsaklis et al. (2018)

Text-to-Entity Mapping: Results

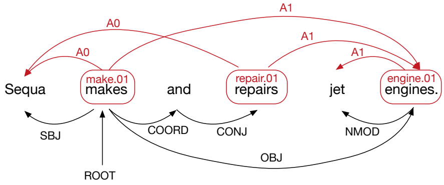
- On the SMOMED CT dataset the text-to-entity mapping outperforms Word2Vec-based baselines
- On reverse dictionary and node classification tasks it shows results comparable to the state-of-the-art techniques (Kartsaklis et al., 2018)

Model	**Target**	**Accuracy**
Baseline	W2V-GoogleNews	0.19
	W2V-PubMed	0.12
MS-LSTM	DeepWalk	0.26
	Enhanced	**0.84**

Source: Kartsaklis et al. (2018)

GCNs for SRL

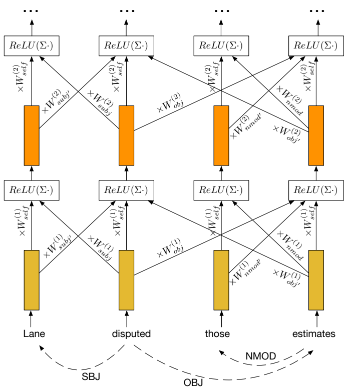
- **Semantic Role Labeling** (SRL) assigns to the words in the sentence the labels corresponding to their semantic role
- Marcheggiani et al. (2017) is the first paper that demonstrates the effectiveness of GCNs for NLP in the SRL setup



Source: Marcheggiani et al. (2017)

GCNs for SRL: Syntactic Dependency Trees

- Syntactic dependency trees are directed, so the layer is $h_v^{(k+1)} = \sigma\left(\sum_{u \in V_v} g_{vu}^{(k)} (V_{\mathrm{dir}(u,v)}^{(k)} h_u^{(k)} + b_{L(u,v)}^{(k)})\right)$
- For each edge-node pair there is a scalar gate: $g_{uv}^{(k)} = \sigma\left(h_u^{(k)} \cdot \hat{v}_{\mathrm{dir}(u,v)}^{(k)} + \hat{b}_{L(u,v)}^{(k)}\right)$

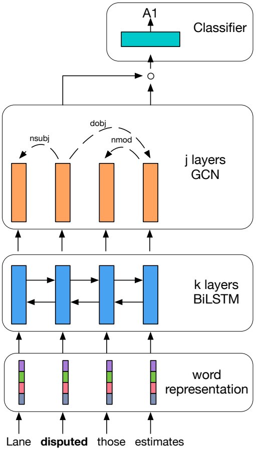


Source: Marcheggiani et al. (2017)

GCNs for SRL: Approach

1. Fetch word embeddings
2. Stack several BiLSTM layers (Hochreiter et al., 1997)
3. Stack several GCN layers (Kipf et al., 2017)
4. Add a softmax classifier

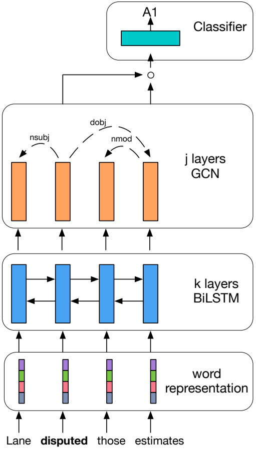
Code and Data: https://github.com/diegma/neural-dep-srl



Source: Marcheggiani et al. (2017)

GCNs for SRL: Results

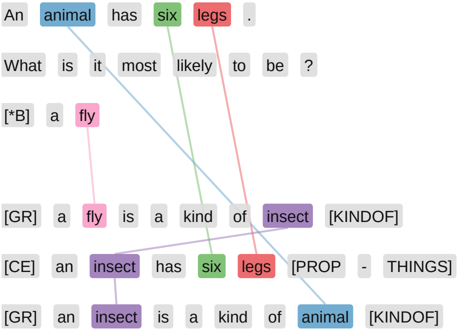
- GCN for SRL outperformed other approaches on both English and Chinese on the CoNLL-2009 dataset
- LSTMs without GCNs outperform GCNs without LSTMs, while their combination dramatically improves the precision
- Even a single GCN layer increases the LSTM-based model accuracy



Source: Marcheggiani et al. (2017)

Explanation Regeneration

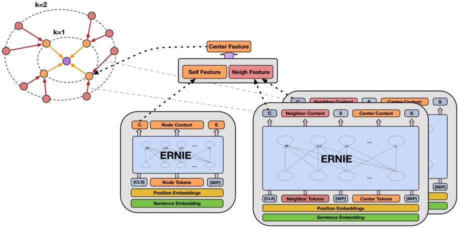
- In the **Explanation Regeneration** task, given an elementary science question with an answer to it, one has to rank explanations of this answer (Jansen et al., 2020)
- The best-performing system at the TextGraphs-14 shared task combined language models and graph neural networks (Li et al., 2020)



Source: Jansen et al. (2020)

Explanation Regeneration: Approach

1. Retrieve the relevant explanations for the questions using ERNIE 2.0 (Sun et al., 2020)
2. Re-rank the retrieved sentences using ERNIE 2.0
3. Aggregate them using the GraphSAGE-like approach (Hamilton et al., 2017)



Source: Li et al. (2020)

Code and Data: `https://github.com/PaddlePaddle/PGL/tree/static_stable/examples/erniesage`

Explanation Regeneration: Example

? A student placed an ice cube on a plate in the sun.
Ten minutes later, only water was on the plate.
Which process caused the ice cube to change to water?
(A) condensation (B) evaporation (C) freezing (D) melting

Rank	Gold	Fact (Table Row)
1	$\star$	melting is a kind of process
2		thawing is similar to melting
3		melting is a kind of phase change
4		melting is when solids are heated above their melting point
5		amount of water in a body of water increases by (storms ; rain ; ice melting)
6		an ice cube is a kind of object
7	$\star$	an ice cube is a kind of solid
8		freezing point is similar to melting point
9		melting point is a property of a (substance ; material)
10		glaciers melting has a negative impact on the glaicial environment

...

Source: Jansen et al. (2020)

Explanation Regeneration: Results

- According to Mean Average Precision (MAP), all the systems have dramatically improved over the tf-idf baseline
- Other systems used BERT, LSTM, integer linear programming, but the best system, BPGL, combined *texts and graphs* (Li et al., 2020)

Model	MAP
tf-idf	0.23
AG	0.37
RDAI	0.55
CSX	0.50
LIIR	0.57
BPGL	**0.60**

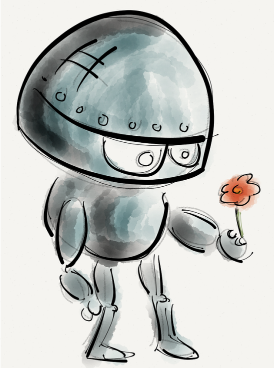
Source: Jansen et al. (2020)

Section 5

Conclusion

Conclusion

- Node embeddings allow incorporating relationships between nodes in a machine learning pipeline
- These techniques improve quality and are available in unsupervised, semi-supervised, and fully supervised setups
- Not covered here: knowledge graph embeddings (Ji et al., 2022), interpretability (Şenel et al., 2018), relationships with BERT-like models (Devlin et al., 2019), expressiveness (Xu et al., 2019)

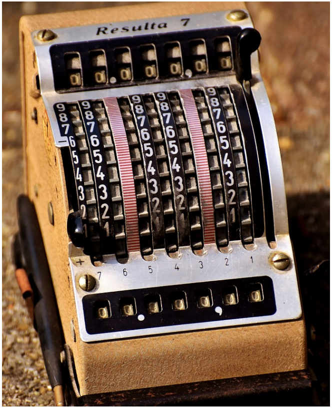


Source: bamenny (2016)

Implementations

- PyTorch Geometric (PyG) (Fey et al., 2019)
- PGL (Ma et al., 2019)
- DGL (Wang et al., 2019)
- GraphGym (You et al., 2020)
- Karate Club (Rozemberczki et al., 2020)

The list is non-exhaustive.



Source: Alexas_Fotos (2017)

Questions?

Contacts

Dr. **Dmitry Ustalov**

https://github.com/dustalov

mailto:dmitry.ustalov@gmail.com

0000-0002-9979-2188

Revision: 5d35748

References I

Belkin M. and Niyogi P. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, vol. 15, no. 6, pp. 1373–1396. DOI: 10.1162/089976603321780317.

Biemann C. and Riedl M. (2013). Text: now in 2D! A framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, vol. 1, no. 1, pp. 55–95. DOI: 10.15398/jlm.v1i1.60.

Bojanowski P. et al. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146. DOI: 10.1162/tacl_a_00051.

Bordes A., Chopra S., and Weston J. (2014). Question Answering with Subgraph Embeddings. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2014. Doha, Qatar: Association for Computational Linguistics, pp. 615–620. DOI: 10.3115/v1/D14-1067.

Brody S., Alon U., and Yahav E. (2022). How Attentive are Graph Attention Networks? *10th International Conference on Learning Representations*. ICLR 2022. Virtual: OpenReview.net. URL: https://openreview.net/forum?id=F72ximsx7C1.

Cai H., Zheng V. W., and Chen-Chuan Chang K. (2018). A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637. DOI: 10.1109/TKDE.2018.2807452.

Chen D. et al. (2020). Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI-20 vol. 34, no. 5, pp. 3438–3445. DOI: 10.1609/aaai.v34i04.5747.

Devlin J. et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. NAACL-HLT 2019. Minneapolis, MN, USA: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

Fey M. and Lenssen J. E. (2019). Fast Graph Representation Learning with PyTorch Geometric. *ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds*. New Orleans, LA, USA. arXiv: 1903.02428 [cs.LG]. URL: https://rlgm.github.io/papers/2.pdf.

Goodfellow I., Bengio Y., and Courville A. (2016). Deep Learning. Cambridge, MA, USA: MIT Press. ISBN: 978-0-262-03561-3. URL: https://www.deeplearningbook.org/.

Goyal P. and Ferrara E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, vol. 151, pp. 78–94. DOI: 10.1016/j.knosys.2018.03.022.

Grover A. and Leskovec J. (2016). node2vec: Scalable Feature Learning for Networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, CA, USA: ACM, pp. 855–864. DOI: 10.1145/2939672.2939754.

References II

Hamilton W. L., Ying R., and Leskovec J. (2017). Inductive Representation Learning on Large Graphs. *Advances in Neural Information Processing Systems 30*. NIPS 2017. Vancouver, BC, Canada: Curran Associates, Inc., pp. 1024–1034. URL: https://proceedings.nips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

Hansen P. C. (1987). The truncated*SVD* as a method for regularization. *BIT Numerical Mathematics*, vol. 27, no. 4, pp. 534–553. DOI: 10.1007/BF01937276.

Hochreiter S. and Schmidhuber J. (1997). Long Short-Term Memory. *Neural Computation*, vol. 9, no. 8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

İrsoy O., Benton A., and Stratos K. (2021). Corrected BOW Performs as well as Skip-gram. *Proceedings of the Second Workshop on Insights from Negative Results in NLP*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 1–8. DOI: 10.18653/v1/2021.insights-1.1.

Jana A. and Goyal P. (2018). Can Network Embedding of Distributional Thesaurus Be Combined with Word Vectors for Better Representation? *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. NAACL-HLT 2018. New Orleans, LA, USA: Association for Computational Linguistics, pp. 463–473. DOI: 10.18653/v1/N18-1043.

Jansen P. and Ustalov D. (2020). TextGraphs 2020 Shared Task on Multi-Hop Inference for Explanation Regeneration. *Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*. Barcelona, Spain (Online): Association for Computational Linguistics, pp. 85–97. DOI: 10.18653/v1/2020.textgraphs-1.10.

Ji S. et al. (2022). A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 494–514. DOI: 10.1109/TNNLS.2021.3070843.

Kartsaklis D., Pilehvar M. T., and Collier N. (2018). Mapping Text to Knowledge Graph Entities using Multi-Sense LSTMs. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2018. Brussels, Belgium: Association for Computational Linguistics, pp. 1959–1970. DOI: 10.18653/v1/D18-1221.

Kipf T. N. and Welling M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *5th International Conference on Learning Representations, Conference Track Proceedings*. ICLR 2017. Toulon, France: OpenReview.net. arXiv: 1609.02907v4 [cs.LG]. URL: https://openreview.net/forum?id=SJU4ayYgl. Licensed under arXiv.org perpetual, non-exclusive license, used with author's permission.

Levy O. and Goldberg Y. (2014). Neural Word Embedding as Implicit Matrix Factorization. *Advances in Neural Information Processing Systems 27*. NIPS 2014. Montréal, QC, Canada: Curran Associates, Inc., pp. 2177–2185. URL: https://proceedings.nips.cc/paper/2014/file/feab05aa91085b7a8012516bc3533958-Paper.pdf.

References III

Li W. et al. (2020). PGL at TextGraphs 2020 Shared Task: Explanation Regeneration using Language and Graph Learning Methods. *Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*. Barcelona, Spain (Online): Association for Computational Linguistics, pp. 98–102. DOI: 10.18653/v1/2020.textgraphs-1.11.

Ma Y. et al. (2019). PaddlePaddle: An Open-Source Deep Learning Platform from Industrial Practice. *Frontiers of Data and Computing*, vol. 1, no. 1, pp. 105–115. DOI: 10.11871/jfdc.issn.2096.742X.2019.01.011.

Marcheggiani D. and Titov I. (2017). Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2017. Copenhagen, Denmark: Association for Computational Linguistics, pp. 1506–1515. DOI: 10.18653/v1/D17-1159.

Mikolov T. et al. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems 26*. NIPS 2013. Lake Tahoe, NV, USA: Curran Associates, Inc., pp. 3111–3119. URL: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Navigli R. and Ponzetto S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, vol. 193, pp. 217–250. DOI: 10.1016/j.artint.2012.07.001.

Ng A., Jordan M., and Weiss Y. (2002). On Spectral Clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems 14*. NIPS 2002. Vancouver, BC, Canada: MIT Press, pp. 846–856. URL: https://proceedings.nips.cc/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf.

Pennington J., Socher R., and Manning C. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2014. Doha, Qatar: Association for Computational Linguistics, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

Perozzi B., Al-Rfou R., and Skiena S. (2014). DeepWalk: Online Learning of Social Representations. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, NY, USA: ACM, pp. 701–710. DOI: 10.1145/2623330.2623732.

Ristoski P. et al. (2018). RDF2Vec: RDF graph embeddings and their applications. *Semantic Web*, pp. 1–32. DOI: 10.3233/SW-180317.

Rozemberczki B., Kiss O., and Sarkar R. (2020). Karate Club: An API Oriented Open-Source Python Framework for Unsupervised Learning on Graphs. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM '20. Virtual Event, Ireland: Association for Computing Machinery, pp. 3125–3132. DOI: 10.1145/3340531.3412757.

Şenel L. K. et al. (2018). Semantic Structure and Interpretability of Word Embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1769–1779. DOI: 10.1109/TASLP.2018.2837384.

References IV

Sun Y. et al. (2020). ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI-20 vol. 34, no. 5, pp. 8968–8975. DOI: 10.1609/aaai.v34i05.6428.

Ustalov D. et al. (2019). Watset: Local-Global Graph Clustering with Applications in Sense and Frame Induction. *Computational Linguistics*, vol. 45, no. 3, pp. 423–479. DOI: 10.1162/COLI_a_00354.

Veličković P. et al. (2018). Graph Attention Networks. *6th International Conference on Learning Representations*. ICLR 2018. Vancouver, BC, Canada: OpenReview.net. URL: https://openreview.net/forum?id=rJXMpikCZ.

Wang M. et al. (2019). Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. arXiv: 1909.01315 [cs.LG].

Wu L. et al. (2018). StarSpace: Embed All The Things! *The Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI-18. Association for the Advancement of Artificial Intelligence, pp. 5569–5577. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11996.

Wu L. et al., eds. (2022). Graph Neural Networks: Foundations, Frontiers, and Applications. Singapore: Springer. ISBN: 978-981-16-6053-5. DOI: 10.1007/978-981-16-6054-2.

Xu K. et al. (2019). How Powerful are Graph Neural Networks? *7th International Conference on Learning Representations, Conference Track Proceedings*. ICLR 2019. New Orleans, LA, USA: OpenReview.net. URL: https://openreview.net/forum?id=ryGs6iA5Km.

Yang C. et al. (2015). Network Representation Learning with Rich Text Information. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*. IJCAI-15. Buenos Aires, Argentina: AAAI Press / International Joint Conferences on Artificial Intelligence, pp. 2111–2117. URL: https://www.ijcai.org/Proceedings/15/Papers/299.pdf.

Yao L., Mao C., and Luo Y. (2019). Graph Convolutional Networks for Text Classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI-19, IAAI-19, EAAI-20 vol. 33, no. 1, pp. 7370–7377. DOI: 10.1609/aaai.v33i01.33017370.

You J., Ying Z., and Leskovec J. (2020). Design Space for Graph Neural Networks. *Advances in Neural Information Processing Systems 33*. NeurIPS 2020. Montréal, QC, Canada: Curran Associates, Inc., pp. 17009–17021. URL: https://proceedings.neurips.cc/paper/2020/file/c5c3d4fe6b2cc463c7d7ecba17cc9de7-Paper.pdf.

Zhong W. et al. (2020). Reasoning Over Semantic-Level Graph for Fact Checking. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. ACL 2020. Online: Association for Computational Linguistics, pp. 6170–6180. DOI: 10.18653/v1/2020.acl-main.549.

Supplementary Media I

Alexas_Fotos (October 7, 2017). Calculating Machine Resulta Old. Pixabay. URL: https://pixabay.com/images/id-2825179/. Licensed under Pixabay License.

Amos E. (December 19, 2011). The Vectrex video game console, shown with controller. Wikimedia Commons. URL: https://commons.wikimedia.org/wiki/File:Vectrex-Console-Set.jpg. Licensed under CC BY-SA 3.0, used with author's permission.

bamenny (February 24, 2016). Robot Flower Technology. Pixabay. URL: https://pixabay.com/images/id-1214536/. Licensed under Pixabay License.

Buissinne S. (August 25, 2016). Dictionary Reference Book Learning. Pixabay. URL: https://pixabay.com/images/id-1619740/. Licensed under Pixabay License.

Finnsson I. (May 19, 2017). Books Covers Book Case. Pixabay. URL: https://pixabay.com/images/id-2321934/. Licensed under Pixabay License.

FreePhotosART (September 3, 2016). Cook Cooking School Pan. Pixabay. URL: https://pixabay.com/images/id-1641959/. Licensed under Pixabay License.

McGuire R. (March 24, 2015). Suit Business Man. Pixabay. URL: https://pixabay.com/images/id-673697/. Licensed under Pixabay License.

Pexels (November 23, 2016). Aquarium Jellyfish Aquatic. Pixabay. URL: https://pixabay.com/images/id-1851643/. Licensed under Pixabay License.

rawpixel (April 18, 2017). Calm Freedom Location. Pixabay. URL: https://pixabay.com/images/id-2218409/. Licensed under Pixabay License.

Simone_ph (March 21, 2017). Music Low Electric Bass. Pixabay. URL: https://pixabay.com/images/id-2149880/. Licensed under Pixabay License.

Wiso (October 24, 2008). Simple neural network. Wikimedia Commons. URL: https://commons.wikimedia.org/wiki/File:Neural_network_example.svg. Licensed under Public Domain.