A GAE Approach to Node Embedding applied to a Link Prediction problem

–

ALTEGRAD

Romy Beaute [1] Kamil Akesbi [2] Grégroire Dutot [3]

[1]ENS Paris-Saclay, France [2]ENS Paris-Saclay, France [3]Ecole Polytechnique, France

21/02/2022

Table of content

Description of the citation problem

Goal : predict wheter a research paper cites another research paper

$\Rightarrow$ Consider this problem as a **link prediction problem** :

Link prediction problem : predicting the existence of a link (citation) between two entities (articles) in a network.

In practice : Compute edge information as a concatenation of node features and use it to train a classifier that predicts the probability of two nodes being linked.

Link prediction in other domains:

- Social Network *(eg. predict common relations or interest)*
- Biology *(eg. predict the function of proteins)*
- Recommender systems *(eg. predict personal taste)*

Idea :

- The messages from some specific neighbors may be more important than messages from others : The GAT layer expands the basic aggregation function of the GCN layer, assigning different importance to each edge. .
- GAT applies self-attention on the nodes : the assigment of importance weights through the attention coefficients

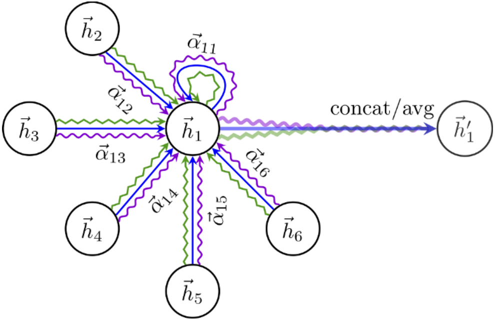

Figure 1: Graph Attention Networks Layer.

GraphSAGE : framework for inductive node embedding.

- The model does not take into account all neighbors of a node, but uniformly samples a fixed-size set of neighbors
- GraphSAGE is used to generate low-dimensional vector representations for nodes
- Useful for graphs that have rich node attribute information
- Capable of predicting embedding of a new node, without requiring a re-training procedure (inductive algorithm)

⇒ GraphSAGE works by learning aggregator functions that can induce the embedding of a new node given its features and neighborhood

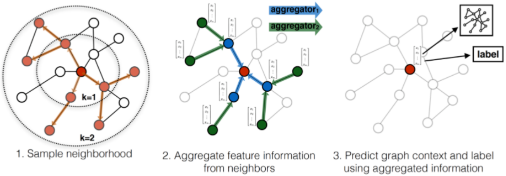

Figure 2: Neighborhood exploration and information sharing in GraphSAGE architecture. *Extracted from http://snap.stanford.edu/graphsage/*

Table of content

Idea : retrieving useful features from the different sources of information

Use of *Unsupervised learning* methods :

- **Citation Graph** : Node2Vec - 50d embeddings
- **Abstracts data** : Doc2Vec - 50d embeddings
- **Authors datas** : Node2Vec on 2 constructed graphs - *32d embeddings*

$\Rightarrow$ Methods based on skip-gram :

$$\min_f \frac{-1}{T} \sum_{i=1}^{T} \sum_{\substack{j=i-w \\ j \neq i}}^{i+w} \log P(v_j | f(v_i))$$

- Difficulty to assess the quality of the embeddings.

We constructed two undirected graphs, learned *32-d* and *64-d* embeddings :

Co-authors graph

- *nodes* : Authors - 174,961 nodes
- *edges* = number of common articles - 569,033 edges

⇒ Average authors features per article

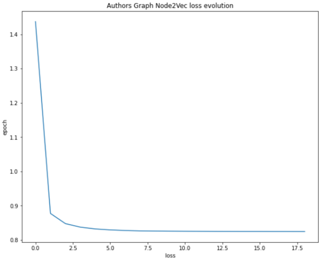

Figure 3: 20 epochs training

Articles linked by commun authors

- *nodes* : articles - 138,499 nodes
- *edges* : number of commun authors - 2,570,106 edges

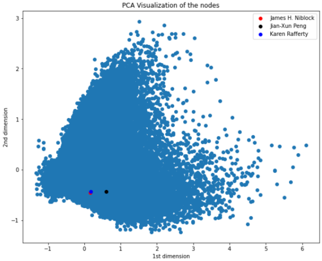

Figure 4: Embeddings visualisation using PCA

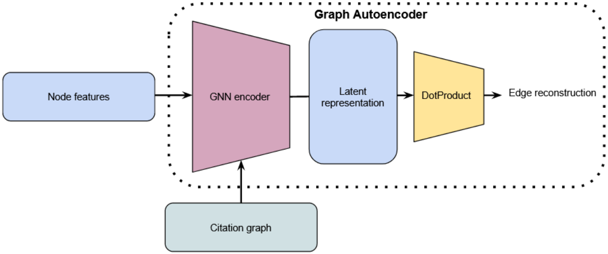

Figure 5: Our graph autoencoder approach

- Edge reconstruction : $\hat{p}_{i,j} = \sigma(Z_i^T Z_j)$;
- Similarity to Link prediction and useful embeddings ;
- An undirected architecture : need for another classifier ;
- Node2vecGrover et al. 2016 initialization of the node features.

- Impossibility to train a classical GNN : we resorted to Message Passing and Neighbor sampling.

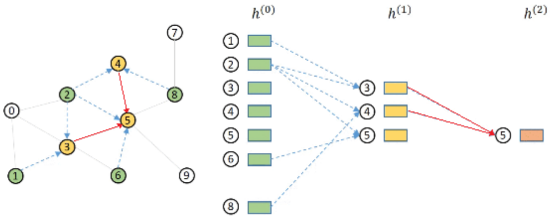

Figure 6: A $[2,2]$-Neighbor sampling strategy used for Message Passing[1]

It gives us :

- More control on memory use ;
- More stochasticity.

We used this strategy for our GraphSAGE (Hamilton et al. 2017) and GAT (Veličković et al. 2018) Message-Passing NN.

[1] https://www.dgl.ai/

Our GAE pipeline

GraphSAGE encoder

- 3 layers ;
- $[7, 15, all_{nodes}]$ Neighbor sampling strategy ;
- Relu activation after the first two layers ;
- 64 hidden features.

GAT encoder

- 2 layers and $[3, 3]$ Neighbor sampling strategy ;
- $z^{(1)} = elu([GAT_1^{(1)}(x^{(0)})|| \dots ||GAT_4^{(1)}(x^{(0)}]))$;
- $z^{(2)} = avg([GAT_1^{(2)}(z^{(1)})|| \dots ||GAT_4^{(2)}(z^{(1)}]))$;
- 64 hidden features.

- Batches of 2048 positive edges and 2048 negative node pairs, from which we compute the message passing graph flows.
- 10 epochs, Adam optimizer on BCE loss.
- At the end of training, we retrieve the embeddings of the encoder for all nodes.

Final Classifier

We have as input features :

- A Doc2vec based representation of the paper's abstract ($\mathbb{R}^{50}$);
- The Node2vec based representation embedding the coauthoring links of the paper ($\mathbb{R}^{32}$);
- A GAE based representation embedding each paper's structural role in the citation network ($\mathbb{R}^{64}$).

We use the concatenation of embeddings for each **directed** pair of node, giving $z_{s,d} = [z_s || z_d] \in \mathbb{R}^{2\times 146}$.

- $z_h^{(1)} = ReLU(W_1 z_{s,d})$, with $W_1 \in \mathbb{R}^{64\times 292}$ & **dropout** with rate 0,5.
- $z_h^{(2)} = ReLU(W_2 z_h^{(1)})$, with $W_2 \in \mathbb{R}^{64\times 64}$ & **dropout** with rate 0,5.
- Final layer : $l_p = W_3 z_h^{(2)}$, with $W_3 \in \mathbb{R}^{1\times 64}$ which gives the log-probability of the v_s being connected to v_d.

Overall architecture

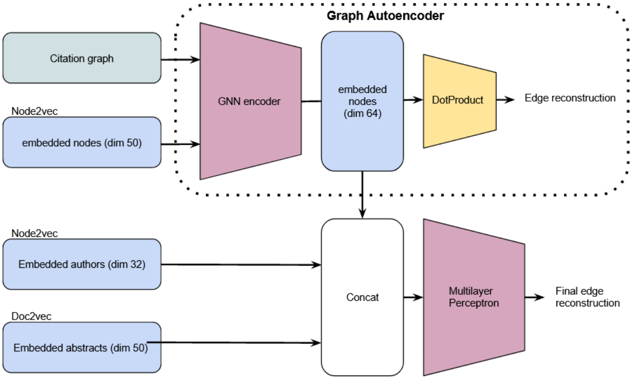

Figure 7: Our whole model pipeline

Table of content

Approach	Train BCE	Val BCE	Test BCE	Time
Graph baseline			0.480	1 min
GraphSAGE based	**0.135**	**0.136**	0.212	18 + 4 min
GAT based (deterministic)	0.093	0.094	0.237	40 + 4 min
GAT based [3, 3]	0.188	0.186	**0.198**	8 + 4 min

Table 1: Scores and computation time for each approach

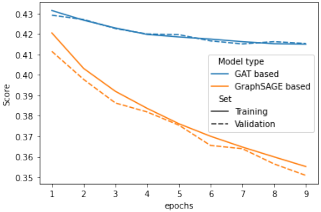

Figure 8: Cross entropy score during the GAE training process

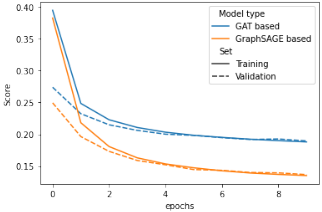

Figure 9: Cross entropy score during the final classifier training process

Table of content

In this project :

- Unsupervised Embeddings on authors, abstracts and citation graphs
- GAE embeddings approach performed better for the link prediction task

Main difficulties

- Lack of transparency of the models
- Great difference between training and test score solved with random sampling strategies

Perspectives

- Training the GAE with all the features

Grover, Aditya et al. (2016). "node2vec: Scalable Feature Learning for Networks". In: *CoRR* abs/1607.00653. arXiv: 1607.00653. URL: http://arxiv.org/abs/1607.00653.

Hamilton, William L. et al. (2017). "Inductive Representation Learning on Large Graphs". In: *CoRR* abs/1706.02216. arXiv: 1706.02216. URL: http://arxiv.org/abs/1706.02216.

Veličković, Petar et al. (2018). *Graph Attention Networks*. arXiv: 1710.10903 [stat.ML].

Thank you!