

This is your last free member-only story this month. [Upgrade](#) for unlimited access.

★ Member-only story

GRAPHEDM-SERIES

Graph Representation Learning — The Encoder-Decoder Model (Part 2)

A deep dive into the GraphEDM architecture



Giuseppe Futia · [Follow](#)

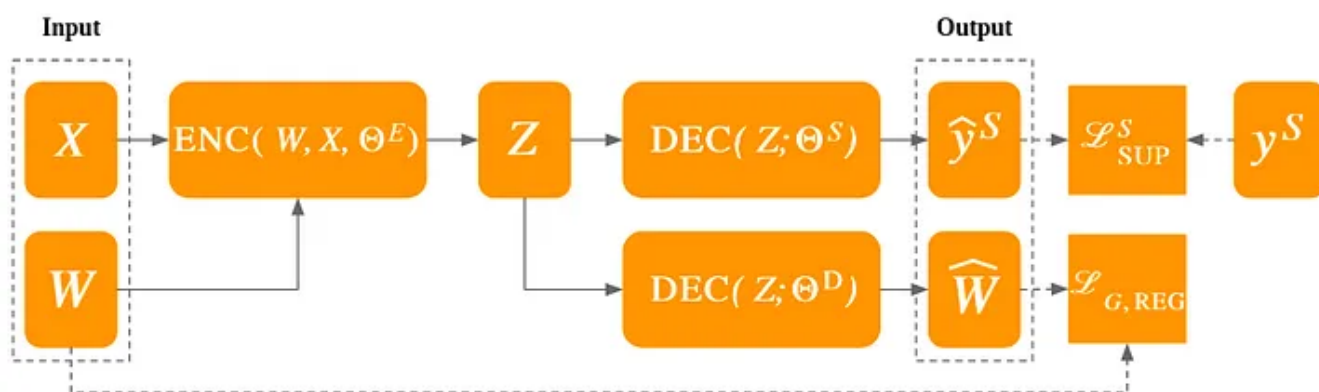
Published in Towards Data Science

6 min read · May 26, 2021

Listen

Share

More



Representation of the GraphEDM framework — Original paper image modified by the author (posted with permission)

This series summarizes a comprehensive taxonomy for machine learning on graphs and reports details on GraphEDM (Chami et. al), a new framework for unifying different learning approaches.

Graphs are ubiquitous structures that are able to encode relations between different elements in diverse domains, from social networks to financial transactions. Graph embeddings are low-dimensional and continuous representations from network-

structured data that can be learned by applying Graph Representation Learning (GRL) techniques.

In the previous article of this series, I proposed a discussion on network embeddings from different perspectives, making different comparisons, including Euclidean vs non-Euclidean geometry, positional vs structural embeddings, and transductive vs inductive learning.

Graph Representation Learning — Network Embeddings (Part 1)

towardsdatascience.com

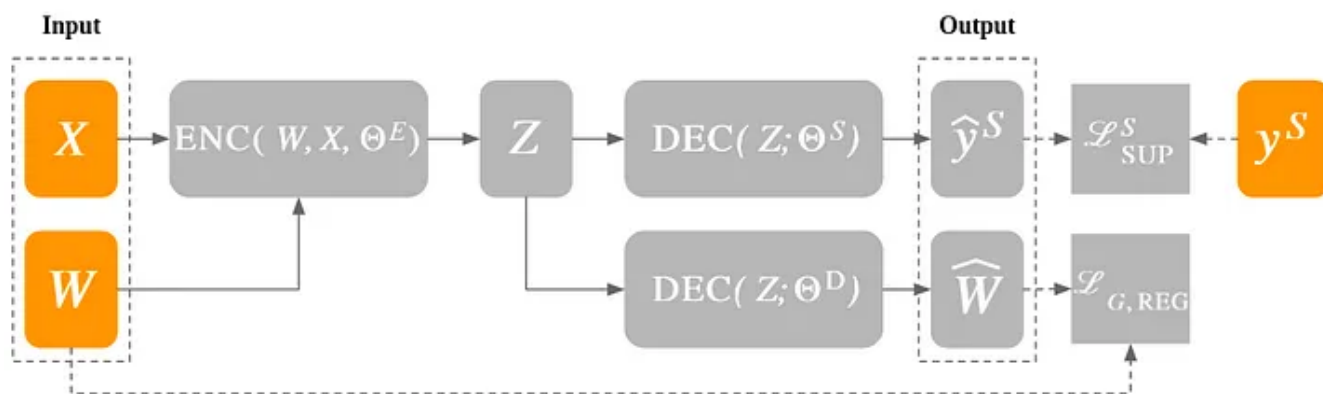
In the second article, we will go into details of the GraphEDM architecture, which coherently comprises different GRL techniques in a unique framework from an encoder-decoder perspective. All the framework details are reported in the article by Chami et. al:

Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., & Murphy, K. (2020). *Machine learning on graphs: A model and comprehensive taxonomy*. *arXiv preprint arXiv:2005.03675*.

The goal of the encoder is to translate the features of a data point into a low-dimensional representation. The decoder takes as input this representation and tries to reconstruct the original data. The performance of this architecture increases if the embeddings computed during the training process are able to capture the core information, which is able to encode the highest variance in the data. Therefore, the learning goal for an encoder-decoder architecture is to minimize information loss during the reconstruction process.

In the case of networks, this reconstruction process intends to restore the original graph structure (unsupervised setting) or to achieve a representation to address a specific task, such as a node or edge classification (supervised setting). Compared to existing research works, GraphEDM defines an end-to-end model that is able to encapsulate both supervised and unsupervised learning methods. The following sections describe the main components of the framework, which is able to generalize more than 30 approaches in the context of GRL.

Input



The input of the GraphEDM framework

[Open in app](#)



The adjacency matrix W is a squared matrix of dimension $|N| \times |N|$ that encodes the relations between the nodes in the graph. In the case of an unweighted graph, W includes values between 0 and 1. Otherwise, for weighted graphs, the values correspond to the edge weights. The number of rows of the X matrix corresponds to the number of nodes in the graph $|N|$, while the number of columns corresponds to the feature dimension d_0 . An illustrative example of such dimensionalities is available in the following article:

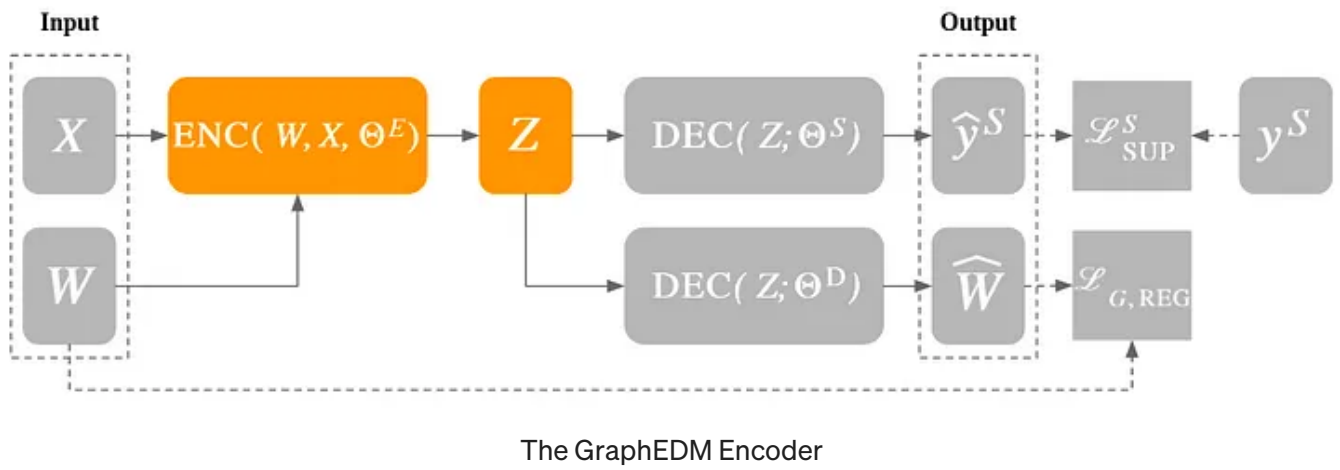
Understanding the Building Blocks of Graph Neural Networks (Intro)

An intuition (with running code) on the framework to analyze and learning graph data with neural architectures

towardsdatascience.com

For the (semi-)supervised setting, we need to include target labels that are used to train the model on a specific downstream task. Training target labels are provided for nodes N , edges E , and/or the entire graph G . The collection of supervision signal is denoted in the GraphEDM paper as $S \in \{N, E, G\}$.

Graph Encoder Network

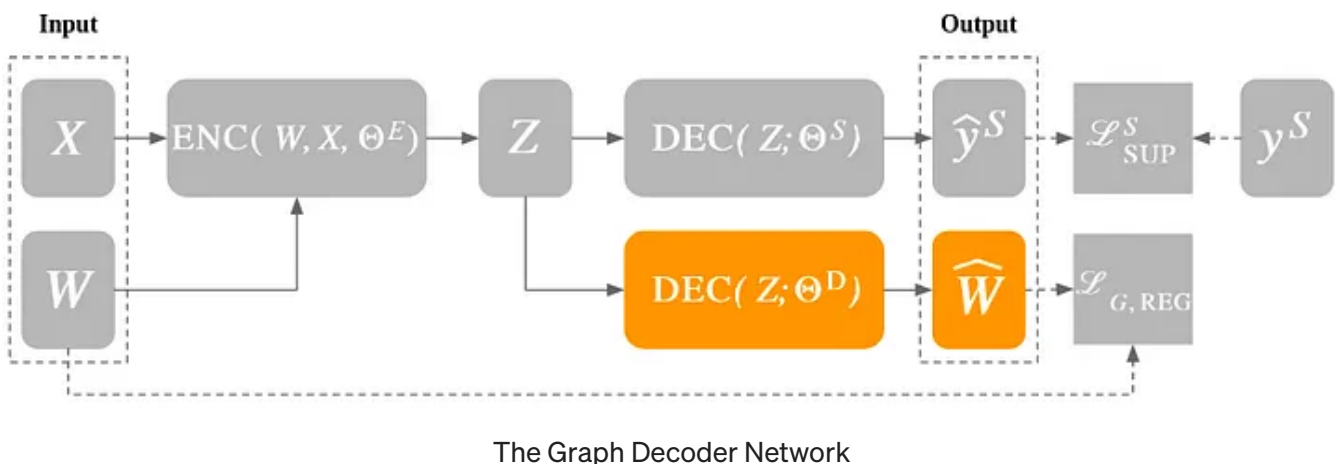


The first component that belongs to the GraphEDM architecture is the Graph Encoder Network, which is denoted as $ENC(W, X; \Theta^E)$. The encoder takes as input:

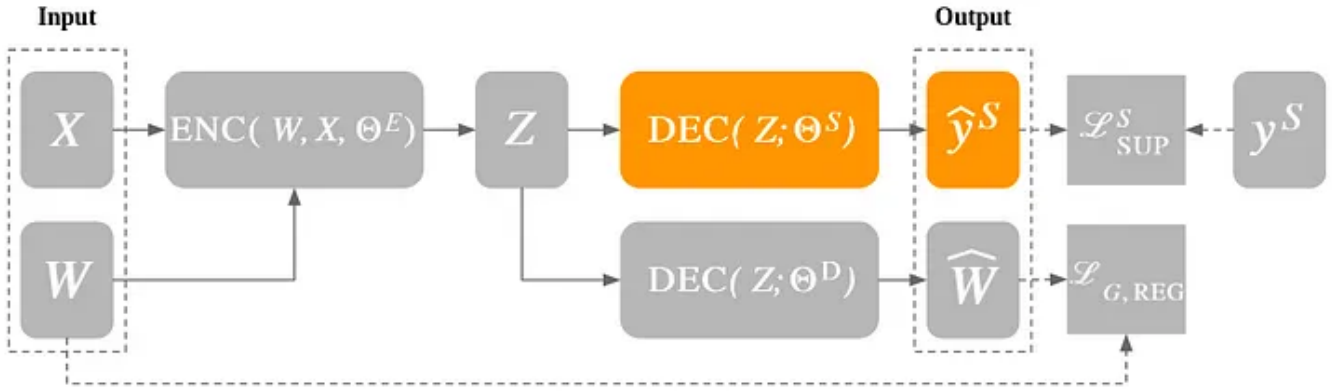
1. the graph structure in the form of the adjacency matrix $W \rightarrow |N| \times |N|$;
2. the node features in the form of the feature matrix $X \rightarrow |N| \times d_0$.

This input is parametrized by Θ^E in order to produce the node embedding matrix $Z = ENC(W, X; \Theta^E)$. The dimensionality of Z is $|N| \times d$, where d corresponds to the node embedding dimension. According to the type of encoder, the node embeddings generated in the training process and encoded in matrix Z might capture different graph properties. For instance, as reported in the previous article of this series, techniques such as matrix factorization and random walk aim at learning low-dimensional representations that preserve the global structure of the input graph. Other techniques, including Graph Neural Networks (GNNs), intend to capture the local graph structure: nodes with a similar local representation should have similar embeddings, in spite of the distance of the nodes in the original network.

Graph Decoder and Classification Network



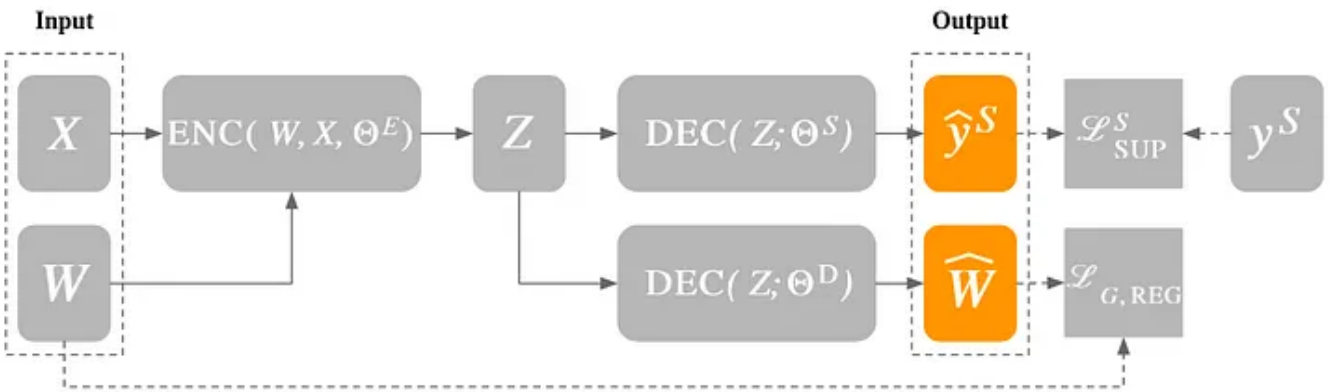
The decoder component $DEC(Z; \Theta)$ takes as input the graph features captured into a low-dimensional representation parametrized by Θ . For unsupervised tasks, the goal of the Graph Decoder Network $\hat{W} = DEC(Z; \Theta^D)$ is to reconstruct the adjacency matrix \hat{W} from the node embeddings Z to compute the similarity (or dissimilarity) scores for all node pairs.



The Classification Network

In the supervised setting, the Classification Network produces label values $\hat{y}^S = DEC(Z; \Theta^S)$ corresponding to an output related to a downstream task. As introduced in the input section, S includes the collection of supervised signals $\{N, E, G\}$.

Output

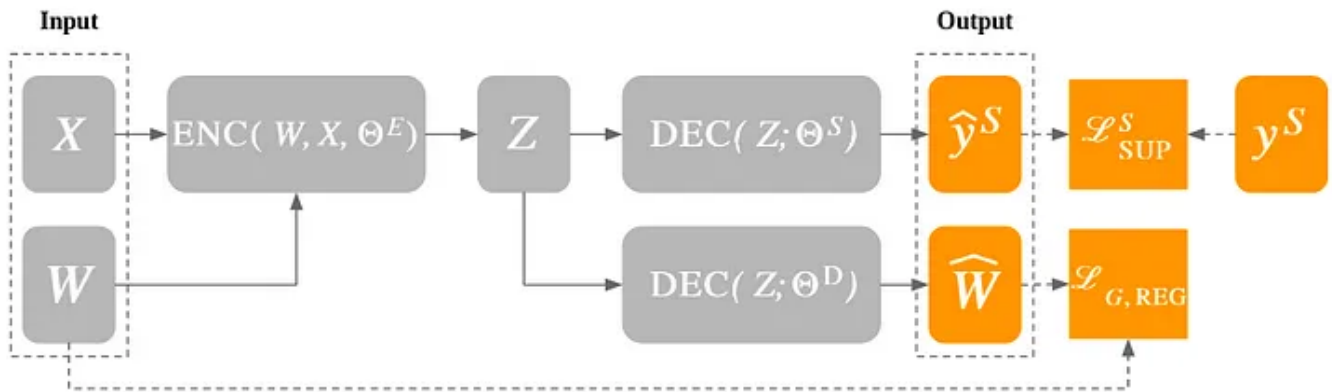


The output of the GraphEDM Framework

The GraphEDM framework is able to produce a twofold output, according to the specific problem setting. Indeed, the output of the model corresponds to an adjacency matrix to train unsupervised embedding algorithms, as well as output labels for the supervised tasks. Intuitively, the dimensionality of \hat{W} is the same of W ($|N| \times |N|$). On the other side, the dimensionality of \hat{y}^S is $|N| \times |Y|$, where Y is the label space. Depending on the supervised application, the label output space can be

different to represent the node label space, the edge label space, and the graph label space.

Loss Functions



Loss functions in the GraphEDM Framework

Different types of loss terms are used to optimize a model in the context of the GRL, including *supervised loss*, *graph regularization loss*, and *weight regularization*.

GraphEDM models are able to combine these terms by learning the parameters mentioned in the previous sections $\Theta \in \{\Theta^E, \Theta^D, \Theta^S\}$.

The supervised loss term L_{sup}^S compares the predicted labels \hat{y}^S with the target labels y^S and it depends on the downstream task (e.g., node classification).

The graph regularization loss term $L_{g, re}$ forces regularization constraints on the weights of the model, leveraging the graph structure. Unlike the supervised loss, the graph regularization compares the decoded adjacency matrix \hat{W} and the ground-truth adjacency W . This type of regularization imposes, for instance, constraints of neighboring nodes to share similar embeddings.

The weight regularization L_{re} traditionally includes a set of techniques that can prevent overfitting in neural architectures, improving their generalization capability. This is enabled learning a regularization coefficient, which defines the constraint boundary in the direction of the minimum loss function. As a consequence, the optimization is prevented to reach the minimum value of the loss function in the training phase, which could lead to overfitting.

What's Next

In the next article of this series, I will provide details on diverse components of the GraphEDM architecture, including the different types of encoders and loss functions.

For all the articles on the GraphEDM framework, you can use the following link:
<https://towardsdatascience.com/tagged/graphedm-series>.

For further readings on Graph Representation Learning, you can follow the related series at: <https://towardsdatascience.com/tagged/grl-series>.

Graphedm Series

Grl Series

Graph Neural Networks

Machine Learning

Graph Learning

Some rights reserved 



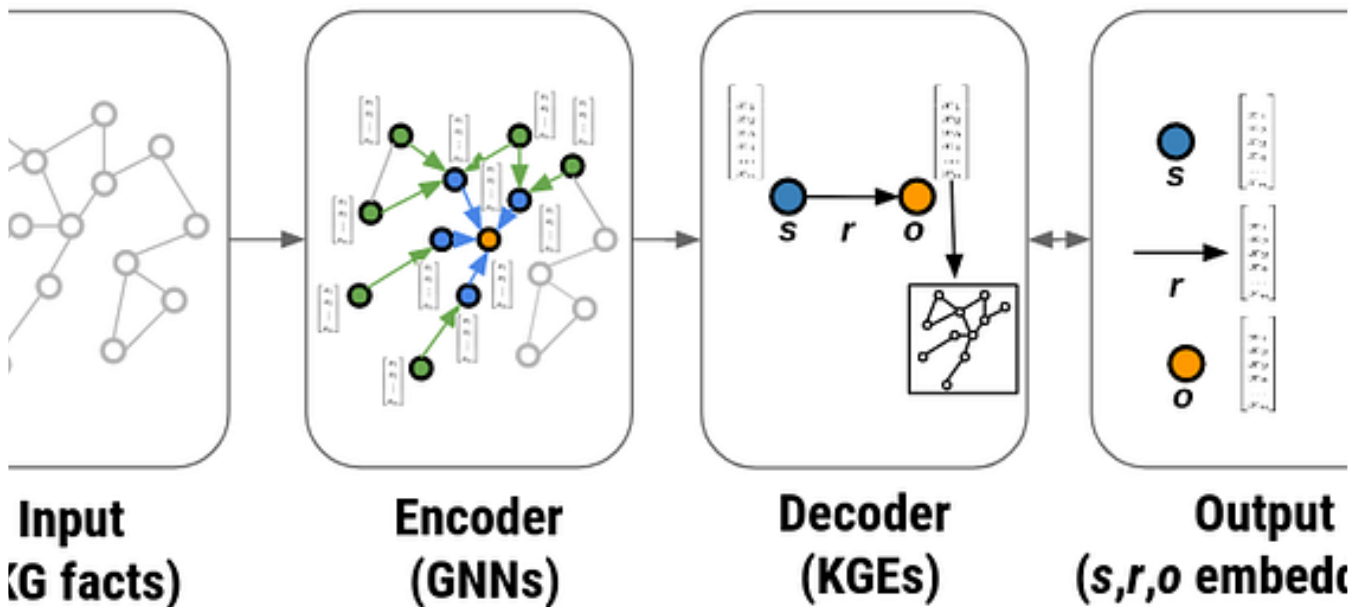
Follow

Written by Giuseppe Futia

750 Followers · Writer for Towards Data Science

Senior Data Scientist at GraphAware | Ph.D. at Politecnico di Torino. Passionate about Knowledge Graphs, Semantic Modeling, and Graph Neural Networks.

More from Giuseppe Futia and Towards Data Science



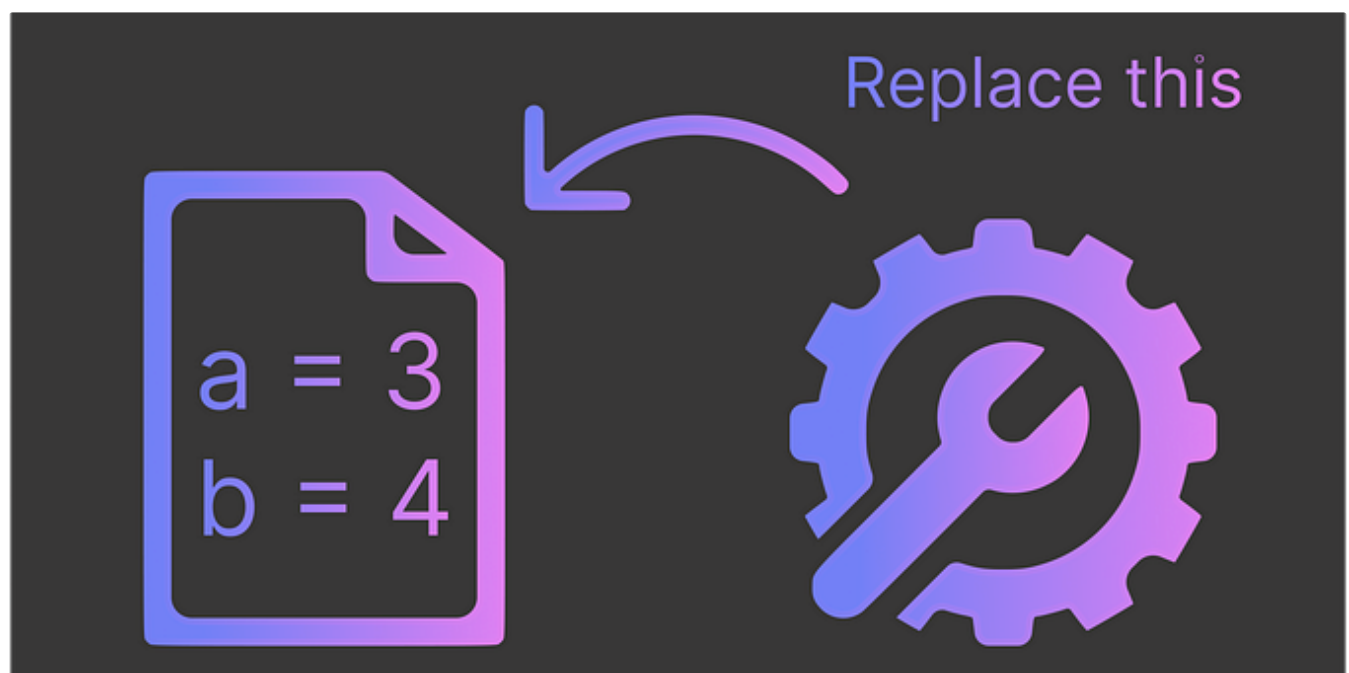
Giuseppe Futia in Towards Data Science

Improving Knowledge Graph Embeddings with Graph Neural Networks

The role of node features derived from the graph structure in the link prediction task

★ · 7 min read · Nov 24, 2020

63 6



Khuyen Tran in Towards Data Science

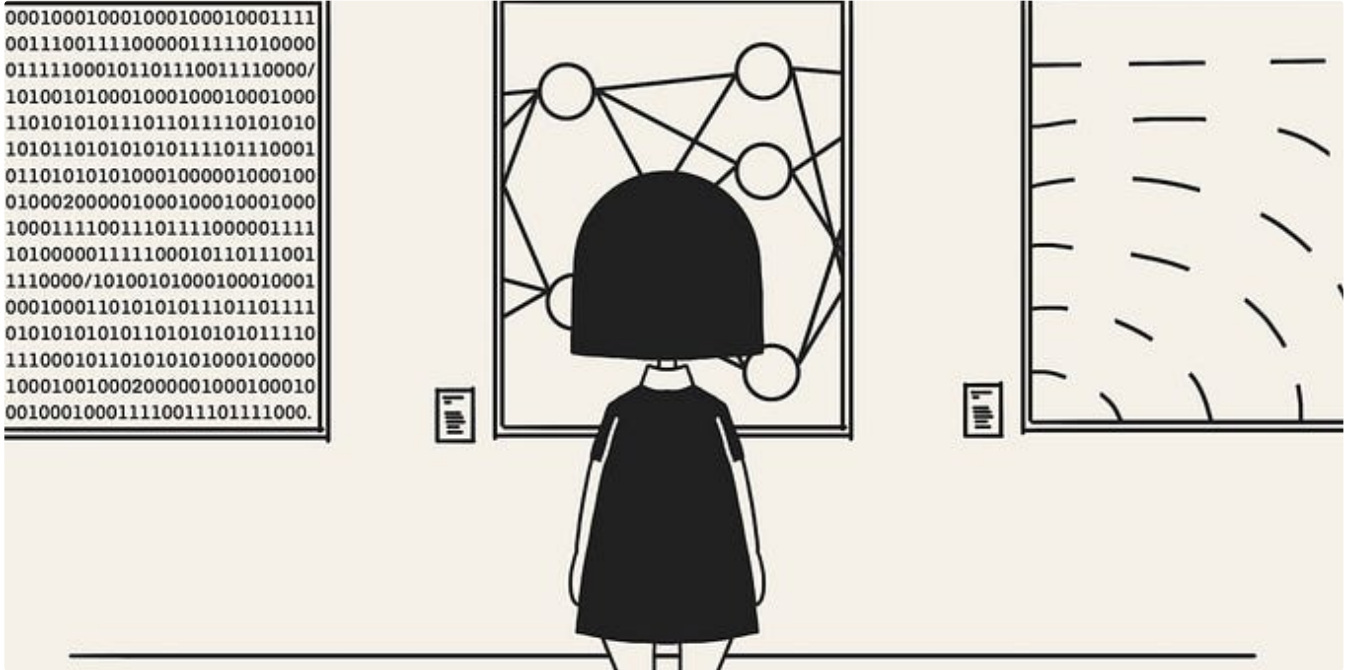
Stop Hard Coding in a Data Science Project—Use Config Files Instead


And How to Efficiently Interact with Config Files in Python

🌟 · 6 min read · May 26

👏 1.8K 💬 19

🔖⁺ ⋮



 Leonie Monigatti in Towards Data Science

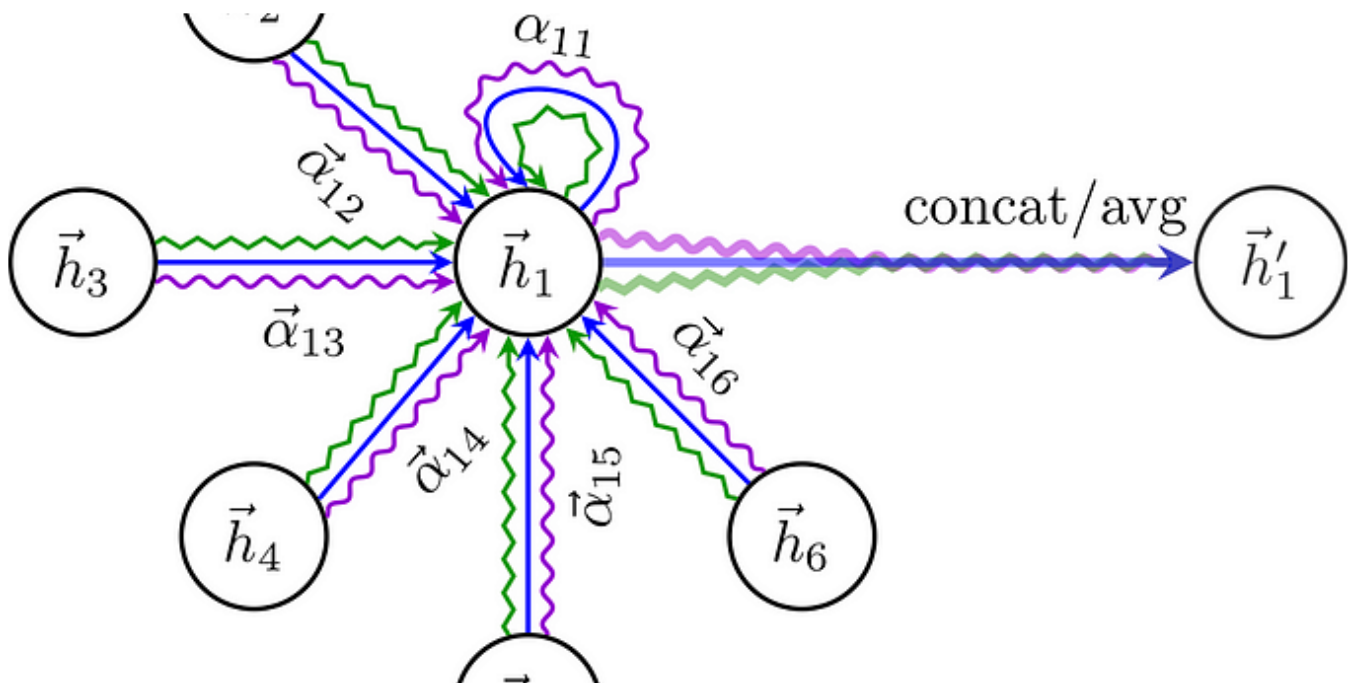
10 Exciting Project Ideas Using Large Language Models (LLMs) for Your Portfolio

Learn how to build apps and showcase your skills with large language models (LLMs). Get started today!

🌟 · 11 min read · May 15

👏 1.6K 💬 10

🔖⁺ ⋮



Giuseppe Futia in Towards Data Science

Graph Attention Networks Under the Hood

A Step-by-step Guide From Math to NumPy



· 8 min read · Feb 1, 2021



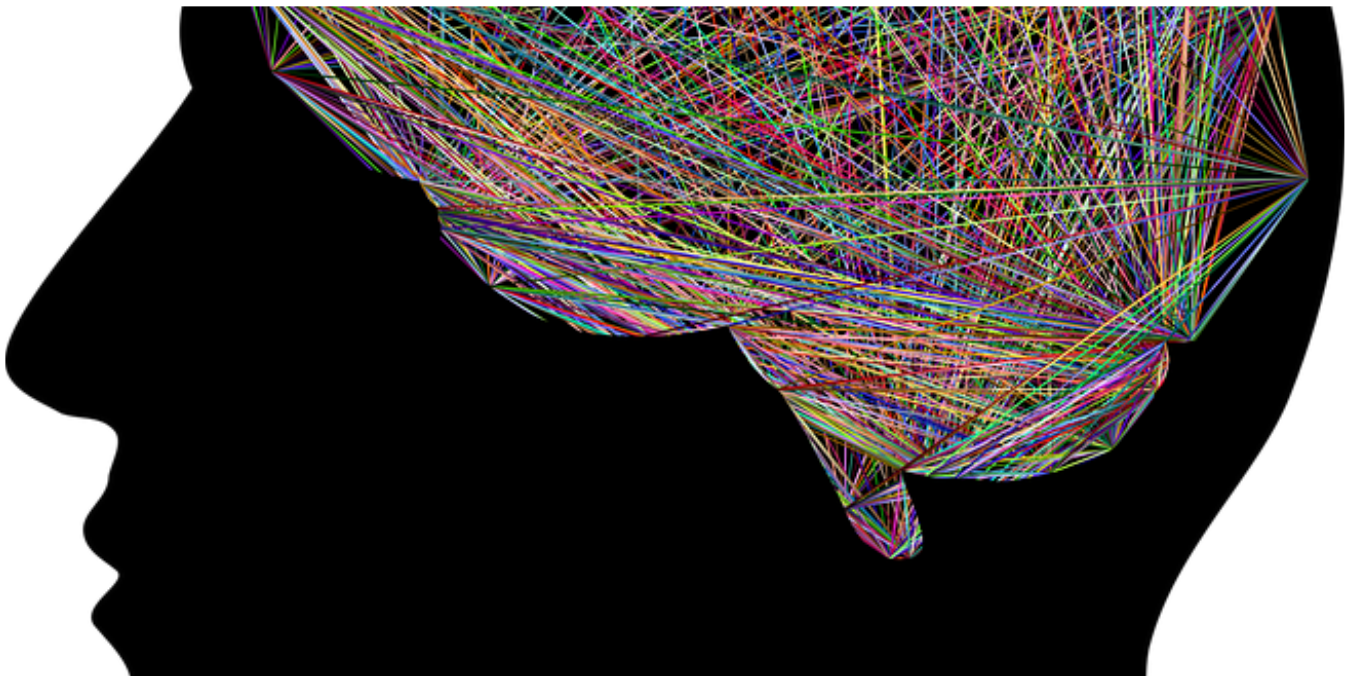
413




See all from Giuseppe Futia

See all from Towards Data Science

Recommended from Medium



 AI TutorMaster in Artificial Intelligence in Plain English

Graph Neural Networks—Introduction for Beginners

A Graph Neural Network (GNN) is a type of neural network that is designed to work with graph-structured data

★ · 10 min read · Jan 14

 424  1



 Conor O'Sullivan in Towards Data Science

Unmasking AI's Detrimental Effects on the Trans Community

The dangers of gender recognition software, inadequate medical models, and the amplification of transphobic content

★ · 8 min read · 6 days ago

👏 226 💬 5



Lists



Predictive Modeling w/ Python

18 stories · 35 saves



Practical Guides to Machine Learning

10 stories · 53 saves



Natural Language Processing

358 stories · 15 saves



The New Chatbots: ChatGPT, Bard, and Beyond

13 stories · 24 saves



Andre Ye in Towards Data Science


Recurrent Neural Networks, Explained and Visualized from the Ground Up

With an application to machine translation

★ · 23 min read · 5 days ago

👏 171



 Ignacio de Gregorio

World Models: The Next Frontier in Our Path to AGI is Here

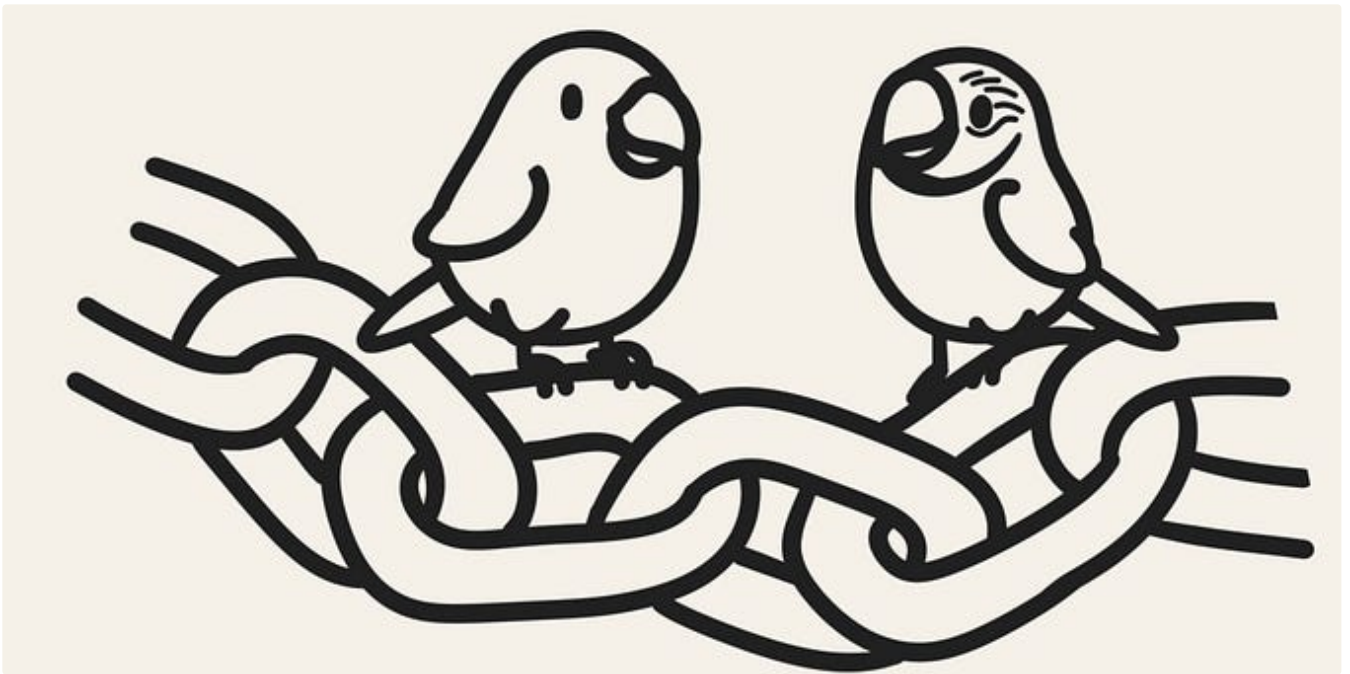
The Common Sense Problem

★ · 8 min read · 4 days ago

👏 560

💬 22






 Leonie Monigatti in Towards Data Science

Getting Started with LangChain: A Beginner's Guide to Building LLM-Powered Applications

A LangChain tutorial to build anything with large language models in Python

🌟 · 12 min read · Apr 25

 3K  19




 Dmytro Nikolaiev (Dimid) in Towards Data Science

Behind the Millions: Estimating the Scale of Large Language Models

Discussing LLMs like ChatGPT, the underlying costs, and inference optimization approaches

★ · 13 min read · Mar 31

 481  6

See more recommendations