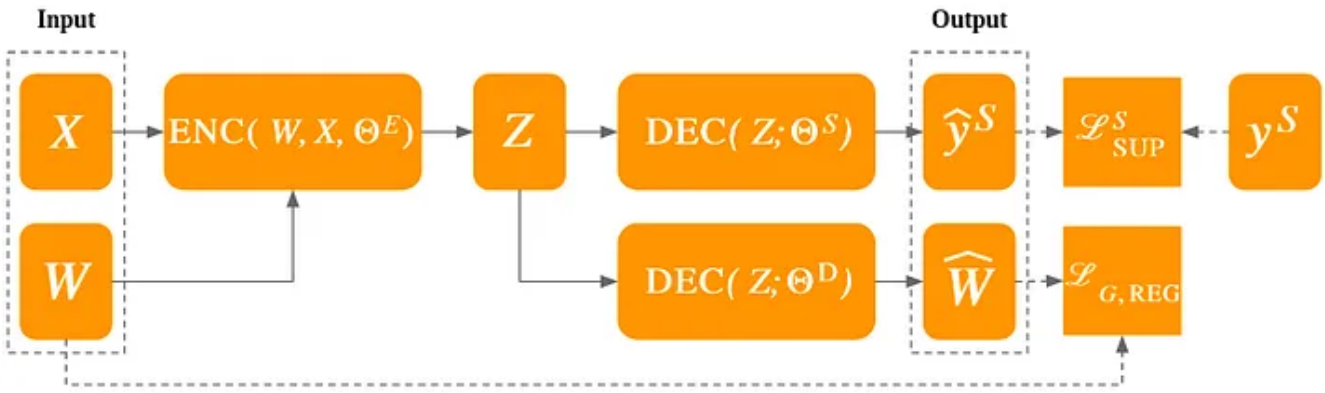
GRAPHEDM-SERIES

# 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






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

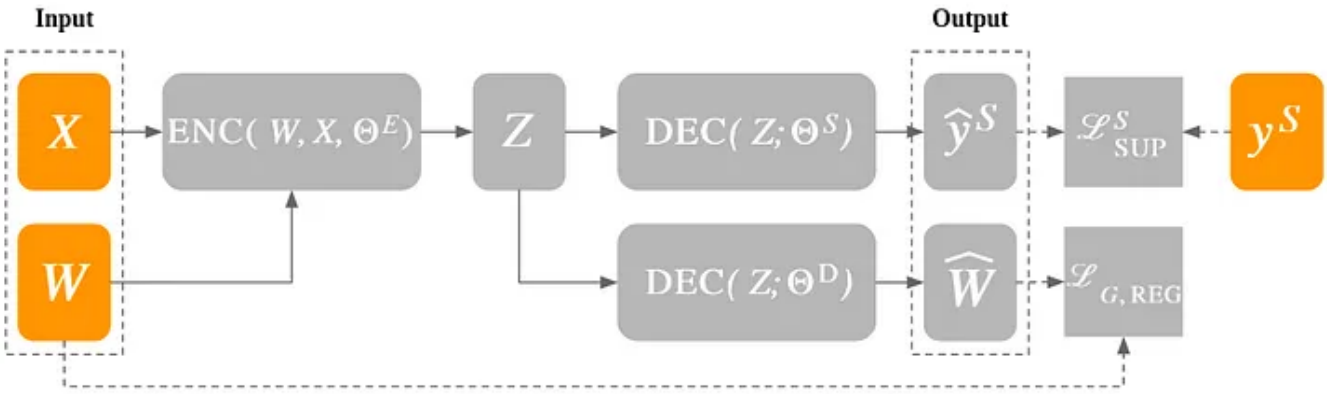
In the second article, we will go into details of the GraphEDM architecture, which coherently comprises different GRL techniques in a unique framework from an encoder-decoder perspective. All the framework details are reported in the article by Chami et. al:

> *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

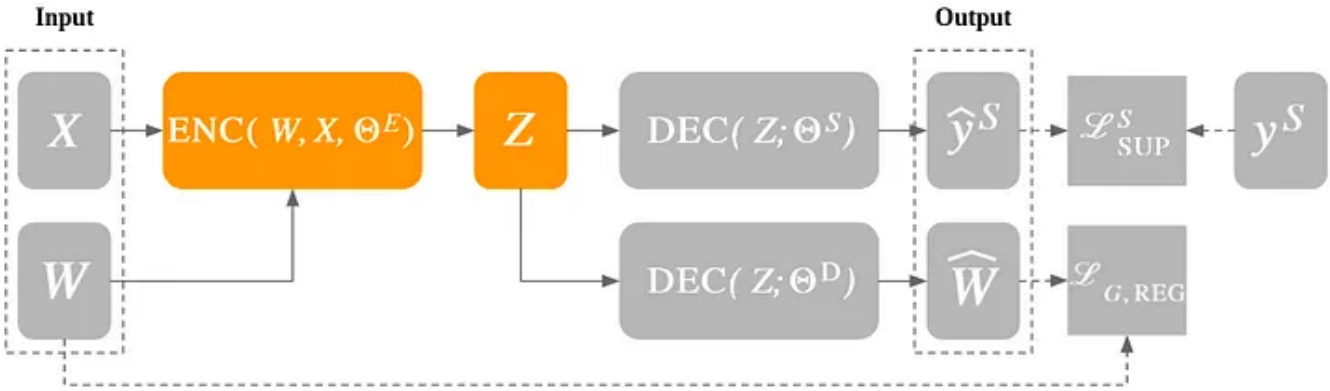
The GraphEDM framework considers as input an undirected and weighted graph, $G=(N, E)$, where $N$ is the set of nodes, while $E$ represents the set of edges. This undirected graph is defined using two different types of matrices: an adjacency matrix $W$ and an optional matrix $X$ describing the node features.

The adjacency matrix $W$ is a squared matrix of dimension $|N| \, x \, |N|$ 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:



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**



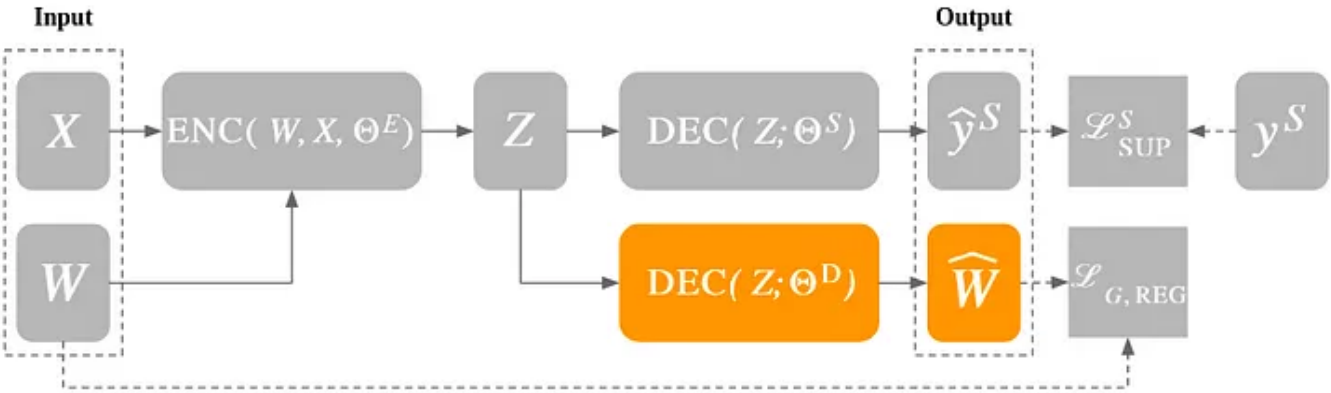

The GraphEDM Encoder

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

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

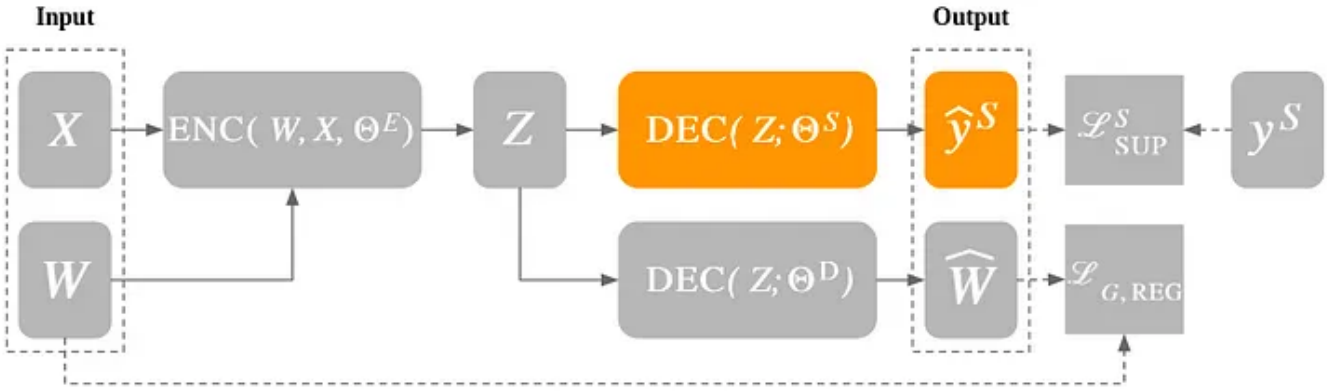
This input is parametrized by *Θ^E* in order to produce the node embedding matrix *Z = ENC(W, X; Θ^E)*. The dimensionality of *Z* is *|N| x d*, where *d* corresponds to the node embedding dimension. According to the type of encoder, the node embeddings generated in the training process and encoded in 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 Graph Decoder Network

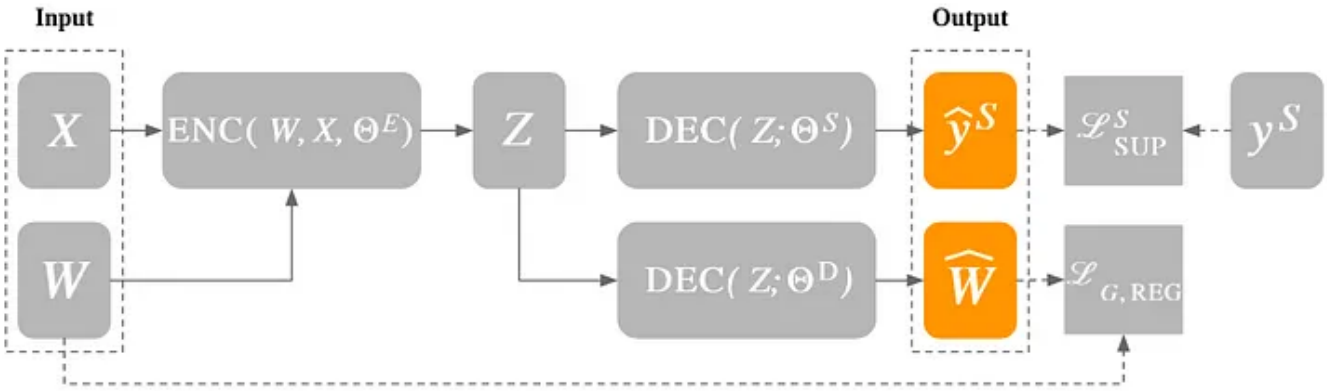
The decoder component *DEC(Z;Θ)* takes as input the graph features captured into a low-dimensional representation parametrized by *Θ*. For unsupervised tasks, the goal of the Graph Decoder Network $\hat{W}$=*DEC(Z;Θ^D)* is to reconstruct the adjacency matrix $\hat{W}$ from the node embeddings *Z* to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

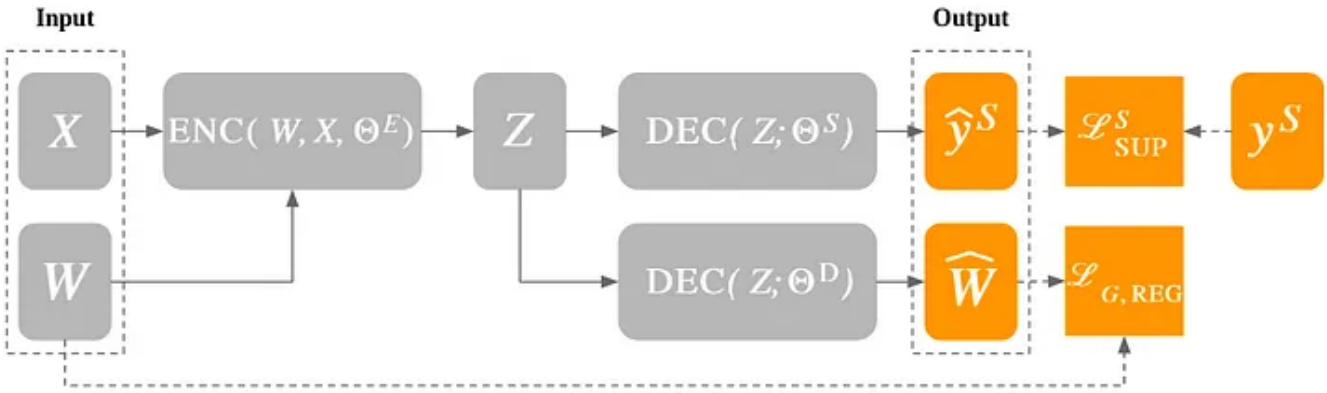



The output of the GraphEDM Framework

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*| *x* |*N*|). On the other side, the dimensionality of $\hat{y}$^*S is* |*N*| *x* |*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 $L_{reg}$ 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 $L_{reg}$ 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



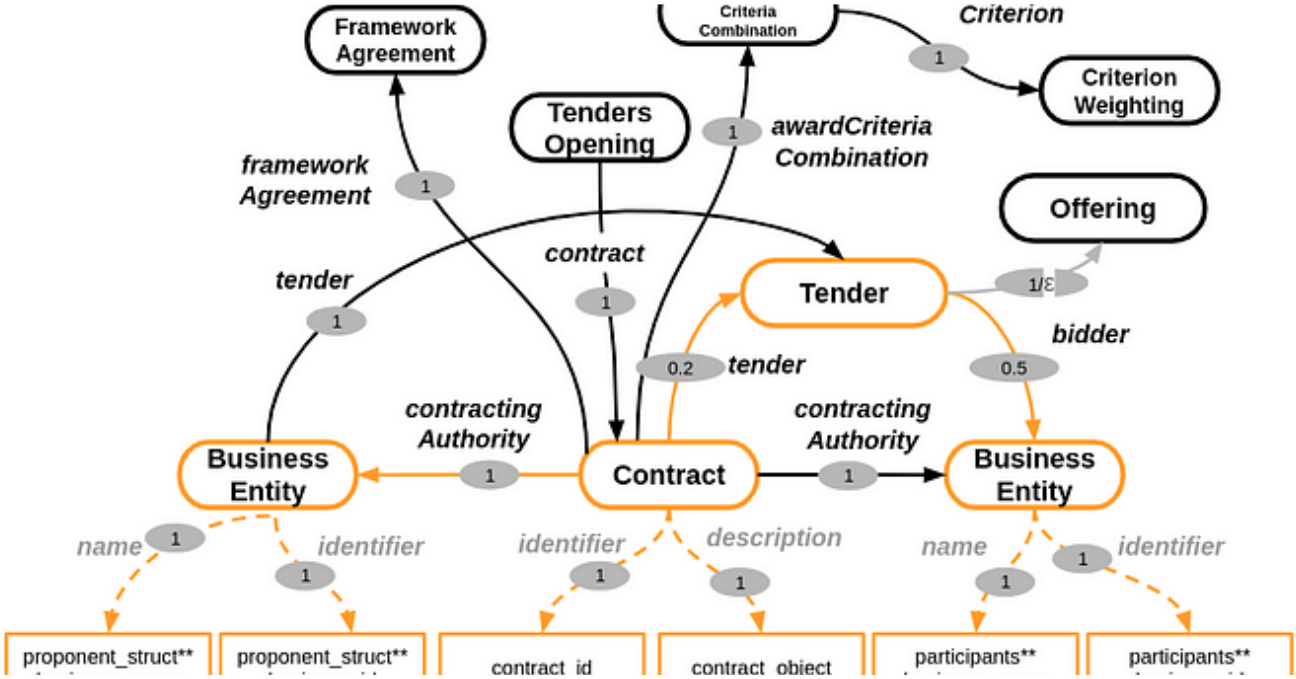
Follow

## Written by Giuseppe Futia

768 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



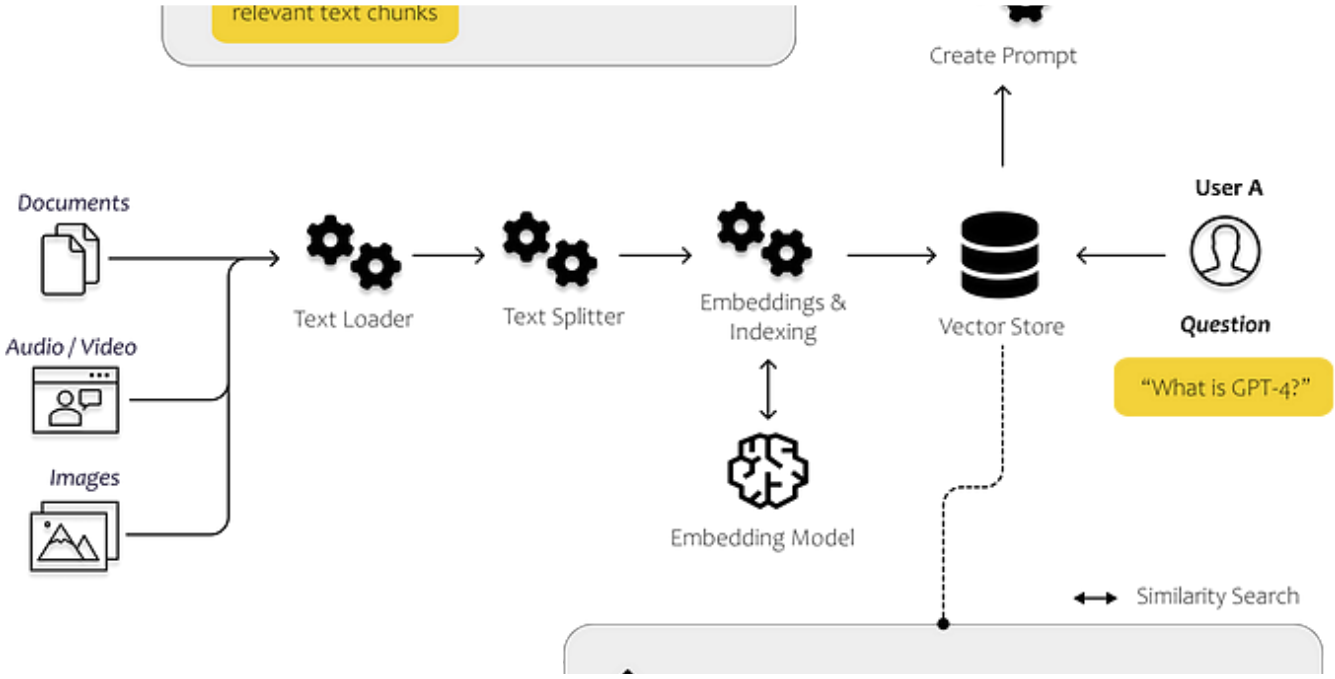

Giuseppe Futia in Towards Data Science

## Semantic Models for Constructing Knowledge Graphs

Capturing the Semantics of Data Sources with Graph-Based Structures

· 6 min read · Dec 16, 2020

69



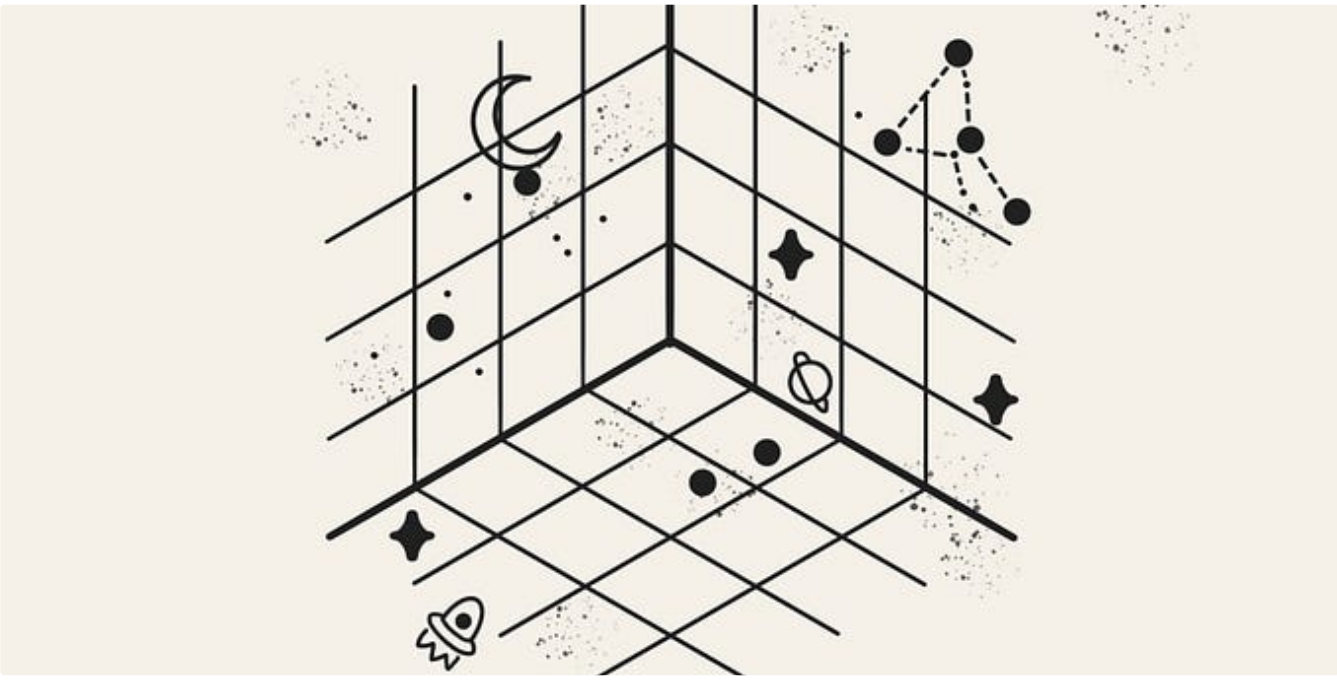

Dominik Polzer in Towards Data Science

## All You Need to Know to Build Your First LLM App

A step-by-step tutorial to document loaders, embeddings, vector stores and prompt templates

✦ · 26 min read · Jun 22

3.4K 31



Leonie Monigatti in Towards Data Science

## Explaining Vector Databases in 3 Levels of Difficulty

From noob to expert: Demystifying vector databases across different backgrounds

✦ · 8 min read · Jul 4

1.8K 18


Giuseppe Futia in Towards Data Science

# Knowledge Graphs at a glance

Incorporate human knowledge into intelligent systems, exploiting a semantic graph perspective

· 7 min read · Sep 27, 2020