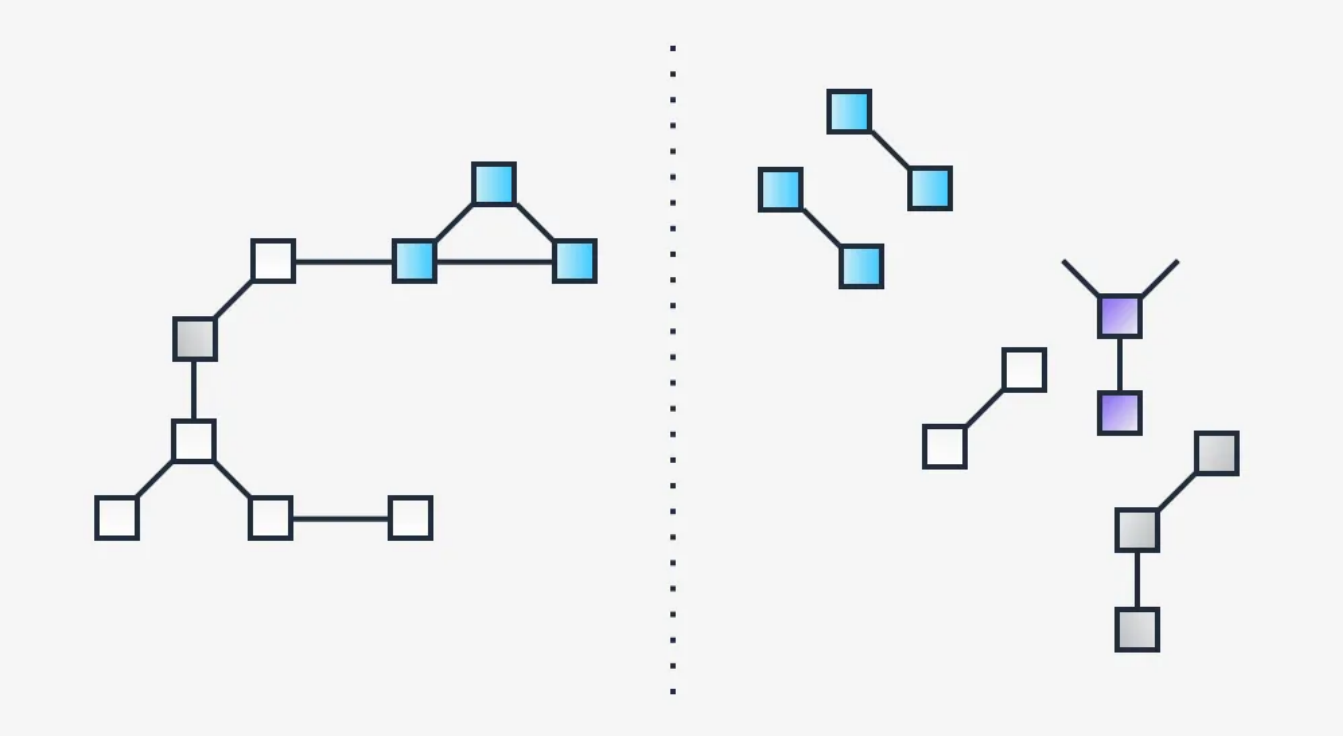
Graphs Part 1: How SingularityNET Will Leverage OpenCog & Aigents

This research series covers how we will be using graphs as data structures and network diagrams to visualize data.

Aigents with Anton Kolonin · Follow
Published in SingularityNET
11 min read · Jun 28, 2018

Listen Share



In this four-part series, we will be looking at different sorts of graphs as data structures and network diagrams used to visualize data. These methods have already applied to SingularityNET partner projects like OpenCog and Aigents.

In this post, **we will provide introductory overview on different types of graphs,** referring to examples rendering these graphs with Aigents graphics framework. Later, we will cover some unique capabilities of using OpenCog in SingularityNET's

infrastructure as a generic graph storage and processing system. Finally, we will also present some basic graph visualization options supported by the Aigents Graphs framework that will also be intergrated into SingularityNET.

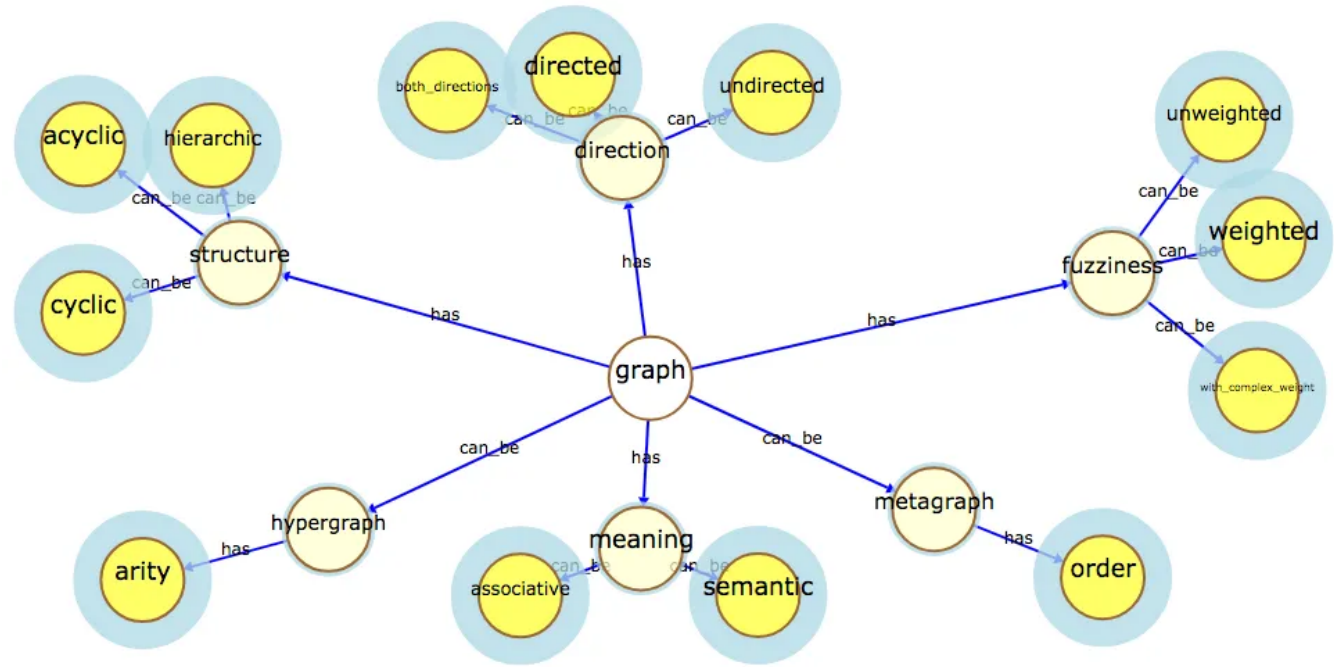
Future posts will describe the use of graphs in the OpenCog Language Learning project and the Aigents News and Reputation agent system, while providing additional use-cases and applications of generalized hyper-graphs and meta-graphs.

Graphs: A Brief Introduction

Graphs can be thought of from at least two perspectives.

1. First, they may represent specific data storage patterns, structures, and solutions like graph databases, triplet stores, specific configurations of relational databases, or systems like OpenCog AtomSpace.

2. Second, graphs may be thought as specific ways to visualize data with various sorts of visual languages, involving nodes or vertices connected by links or edges, like the visual language used by the Aigents Graphs framework.



Graphs ontology rendered with Aigents Graphs

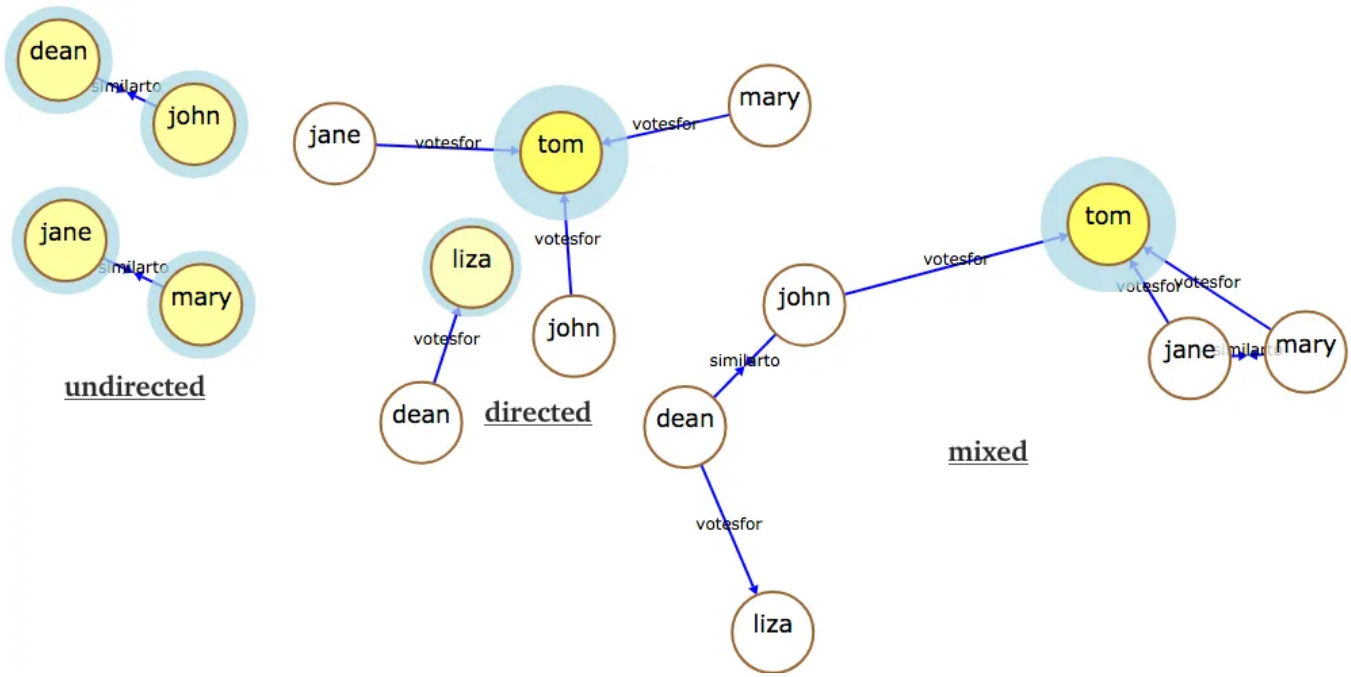
Let's recap the different ways we can differentiate the many types of graphs, according to the roadmap (or road-graph) rendered above. We will be referring to nodes connected by links. All examples will be illustrated with web links to real graphs rendered with the Aigents Graphs framework.

Exploring the Many Types of Graphs

Links may have **direction,** so they may be **directed** or **undirected,** so the graph containing undirected links only can be **undirected**. A graph consisting of directed links may be called **directed**.

Undirected links are symmetric links connecting two nodes, like a link between two people. Directed links imply asymmetry, so if one person votes for another, it does not necessarily mean the latter is voting for the former.

Moreover, a graph may be thought as having **mixed** direction if it has both directed and undirected links.



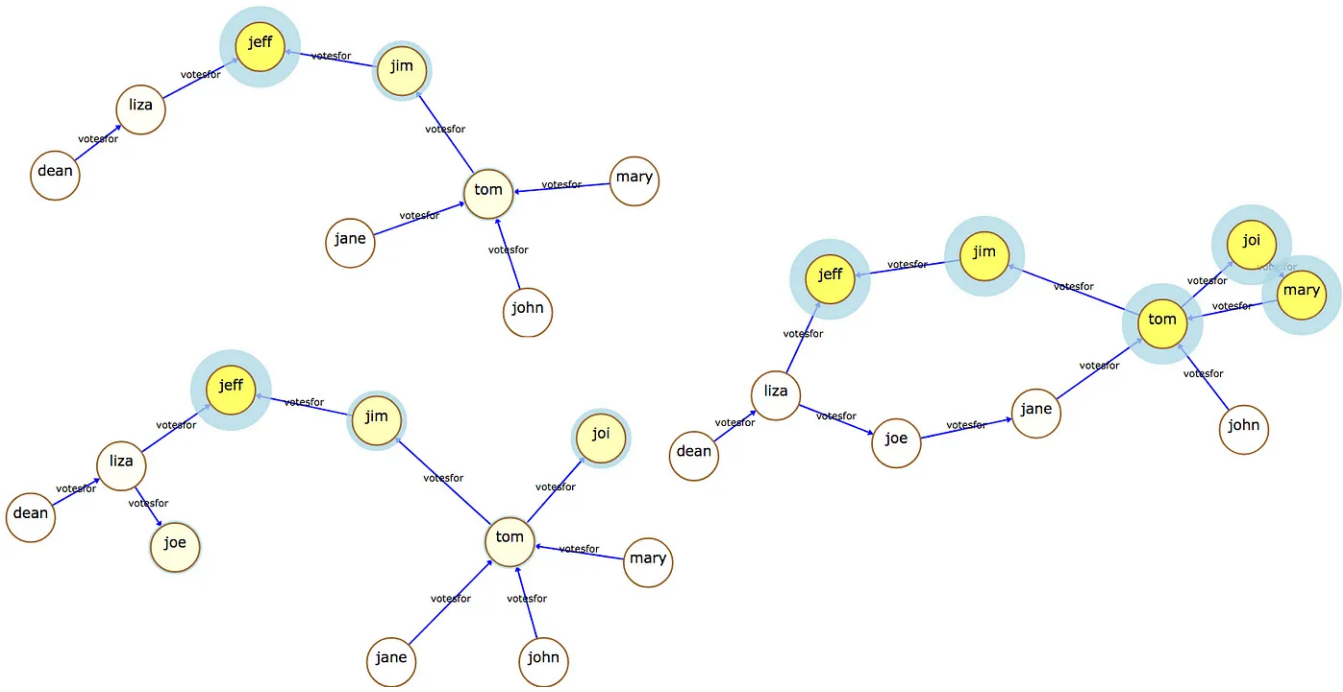
Undirected, directed and mixed direction graphs rendered with Aigents Graphs

Regardless of direction, graphs may have a **structure** with different levels of complexity. For example, there is a **tree or acyclic graph**, defined so that any two nodes in the graph may be connected only with one path (i.e. only one unique chain of links). An example of such tree is a graph involving children and parents.

A specific version of the tree is a **rooted tree or hierarchical graph**, where the directed link paths from any node in the graph lead to only one **root.** An example of such a structure is a subordination chart in the army.

Additionally, there are **cyclic graphs,** where multiple link paths may lead from some nodes to others, making it possible to go in circles along the paths. An example is a friendship relationship chart between many people. Note that some nodes

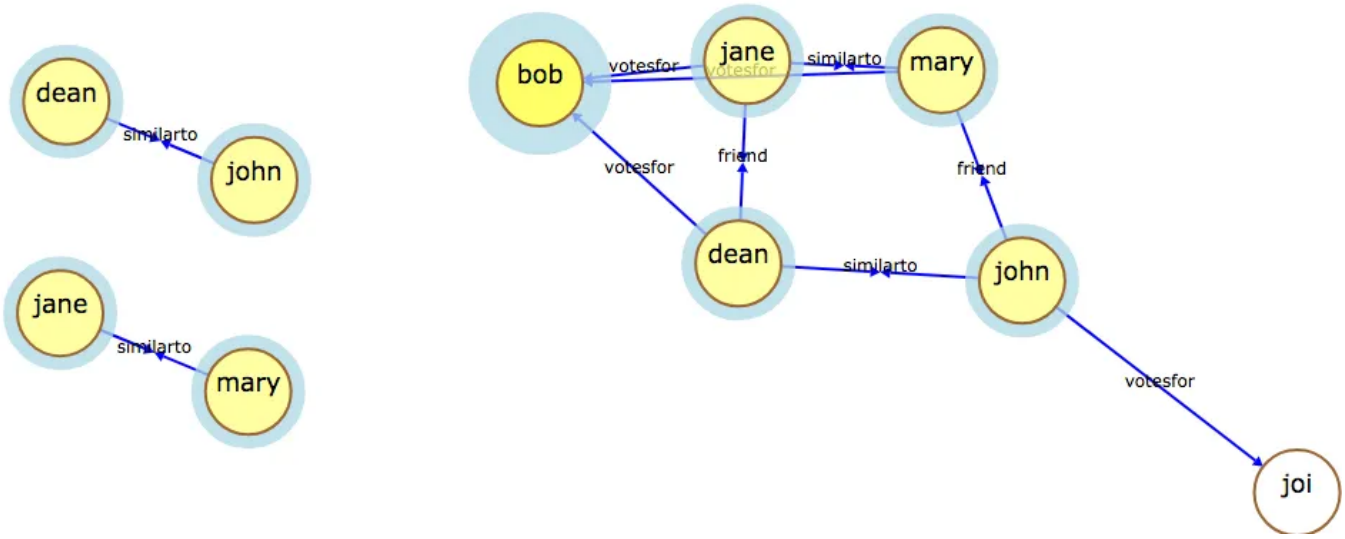
concentrating unique and non-unique paths from different links and different segments of a graph are called **hubs** or **concentrators**.



Rooted tree or hierarchical graph (left-top), tree or acyclic graph (left-bottom) and cyclic graph (right) rendered with Aigents Graphs

Links on a graph may have one or more **meanings**, expressing only one idea or multiple ideas at the same time. In a simple case, a graph may be **associative** or have only one kind of links expressing a particular kind of association between the nodes on graph, like mutual similarity. In more complex cases, multiple kinds of links with different meanings may be present in the same **semantic** graph, building a so called **semantic network**, corresponding to some **ontology**.

In the latter case, the graph may also be called a **labeled** graph, because different types of links have to be differentiated in some way such as attaching different labels to different kinds of links. This kind of graph is widely used in products and projects implementing so-called **Semantic Web**, having these graphs stored in so-called **triplet stores** or conventional **graph databases**.



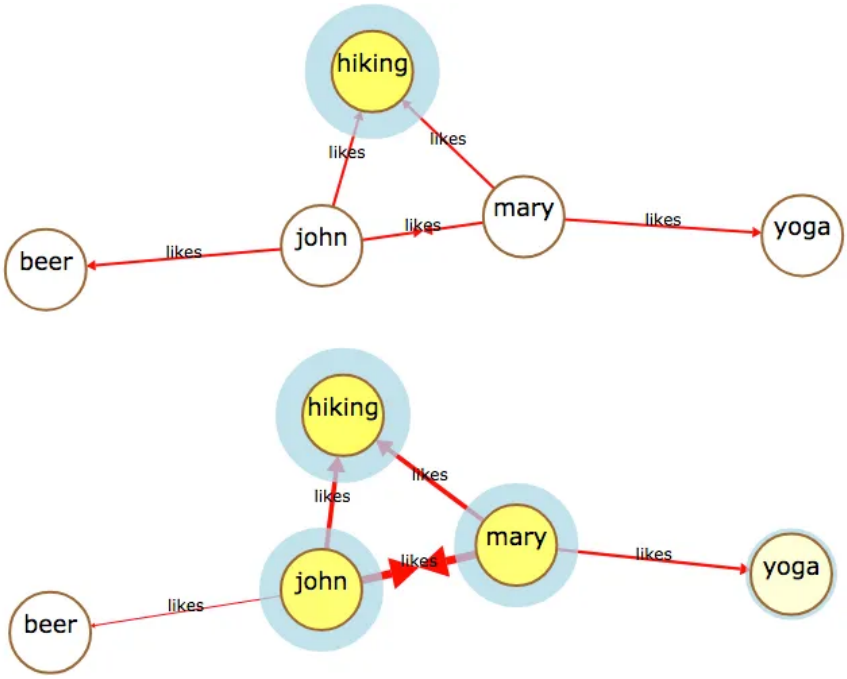
Simple associative (one meaning)and complex semantic (many meanings) graphs rendered with Aigents Graphs

A graph may be considered to have a particular degree of **fuzziness**, corresponding to the interpretation of the link value between the nodes. The most simple graph is an **unweighted** non-fuzzy graph, simply stating the presence of some relationships between subjects and objects and not assessing values of these relationships. A useful example of such a graph is a genealogic tree.

A more complex form of the graph is a **weighted** fuzzy graph, with some value attached to every link indicating the weight of the links as the strength of a corresponding relationship or probability of its existence. The interpretation and numeric scale of such a weight may vary from case to case, but it's often considered in ranges like 0–100% or 0.0–1.0. A good example of such a graph is storing the preferences of people where some connections between subjects and objects may be more important and valuable than others.

Finally, there is an even more complex case with **complexly weighted** graphs, where each link is supplied with a composite **truth value**, which may consist of the evidence count of the relationship observed between entities represented by nodes, the inferred strength of the relationship, and the confidence of the inference. An example of the latter is a connection between thunder and rain, which may have high positive strength but low confidence if observed in a deserted area only a few times, and high confidence if inferred from observations in the tropics with a high volume of evidence.

In the most complex case, each of the links may be associated with a complete probability distribution inferred from the log of supplied evidence.

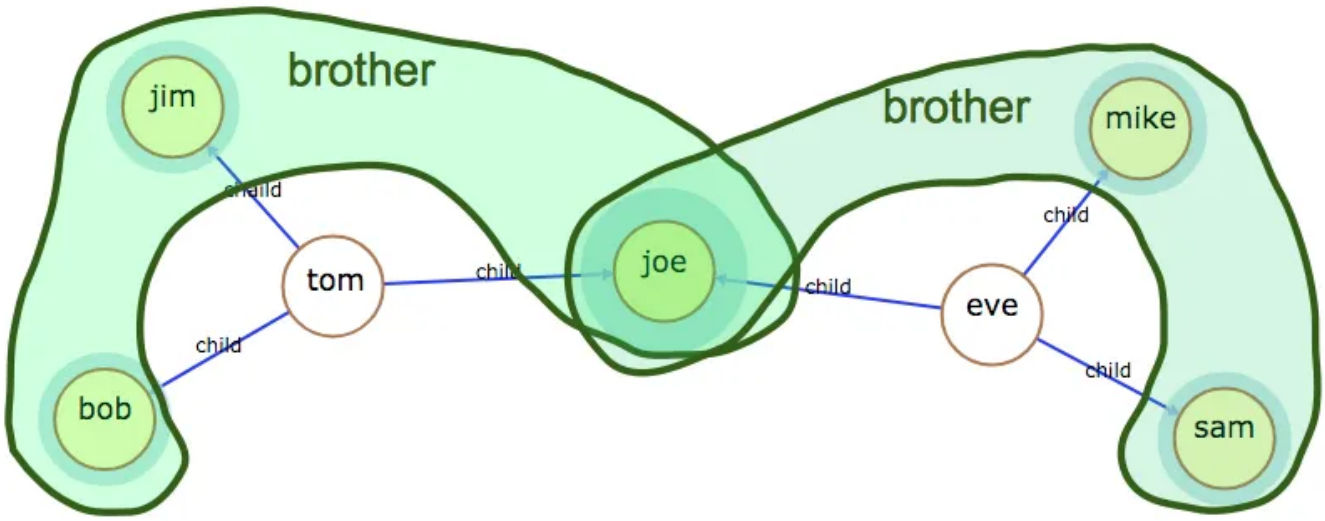


Unweighted (top) and weighted (bottom) graphs rendered with Aigents Graphs, with greater weight represented with greater width of arrows representing the links between the nodes

A more complex kind of graph, going beyond plain nodes and links between them, is a **hyper-graph**, where more than one node may be aggregated with a single link, called a **hyper-link**. In such case, the amount of nodes in the link correspond to link **arity**, so that a conventional binary link connecting two nodes has an arity of 2, while a ternary link connecting three nodes has an arity of 3.

A good example of such a graph is a kinship graph where all brothers and all sisters may be connected by links collecting them as a set. In this case the set would be an undirected link. Another example could be the inheritance of a throne graph with all rulers of a country ordered in a predecessor-to-successor series, being single ordered hyper-link.

It should be noted that each hyper-graph can be theoretically represented with a more atomic plain graph consisting of a larger number of nodes and binary links. For instance, a single “brother” hyper-link connecting three brothers may be represented with a node “common parent” and three “son” links.



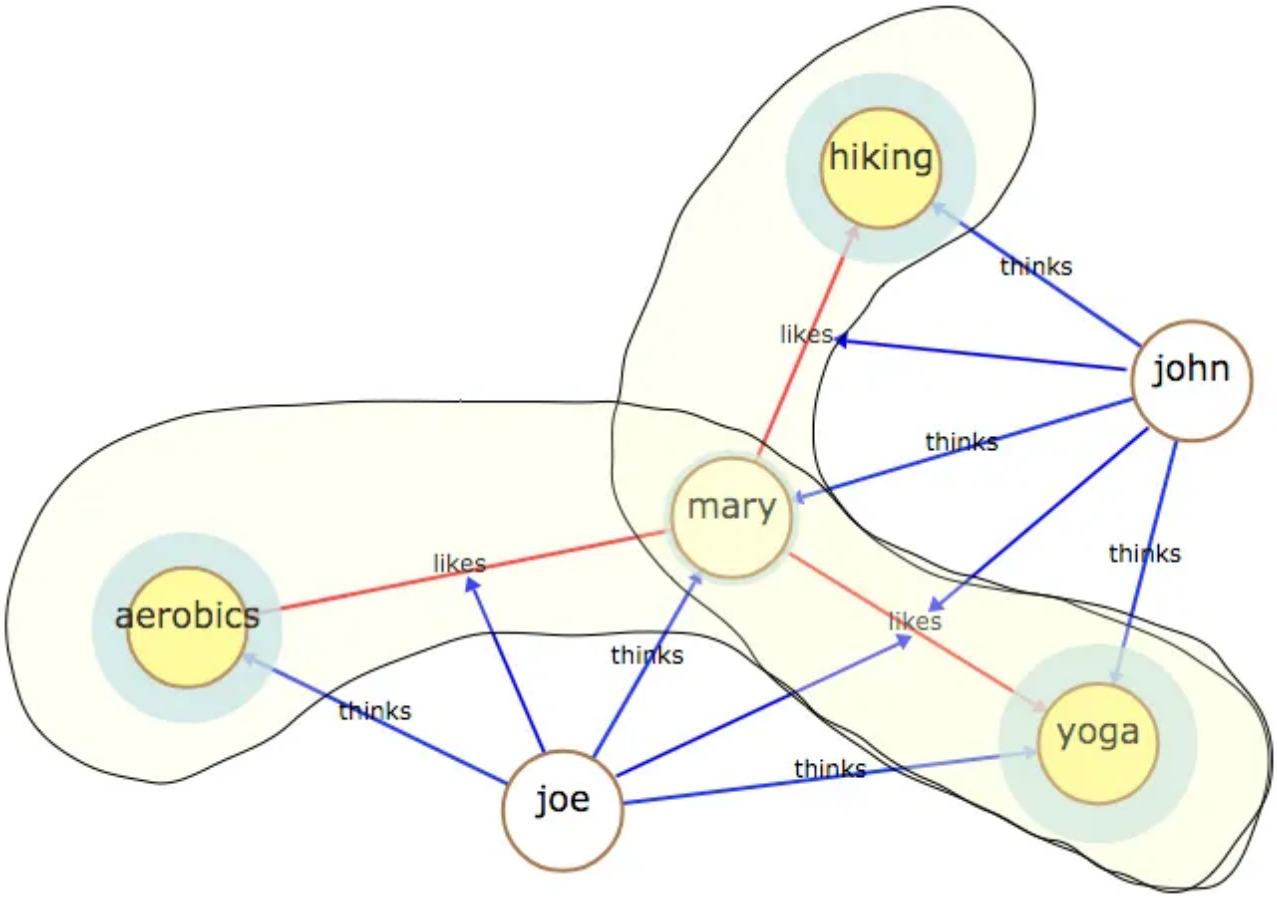
Example of hyper-graph overlaid on plain graph rendered with Aigents Graph — Jim, Bob and Joe are all linked by 3-ary “brother” link because of common father Tom, while Joe, Mike and Sam are linked by another 3-ary “brother” link because of common mother Eve.

An even more complicated form of the graph is a **meta-graph** (or **higher-order graph**), which makes it possible for one graph to play the role of a node in another graph, which is called a **sub-graph,** while the referring outer graph is called a **super-graph.**

In meta-graphs, super-graphs may be enclosing sub-graphs iteratively and the nesting level may be called an **order**, so graphs of a lower order, being sub-graphs, are enclosed in graphs of a higher order, being super-graphs.

A basic example of such situation is the expression of beliefs of different subjects referring to the same shared objects but involving different links between these objects and the others. For instance, both John and Joe may have done yoga with Mary, however John also experienced Mary hiking while Joe knows Mary as involved in aerobics — so both guys have different internal images of Mary, overlapped to some extent. This can be represented by having each of the two different subjective experiences recorded in two separate sub-graphs, and by having a higher-order super-graph involving Joe, John, and their belief connected to them.

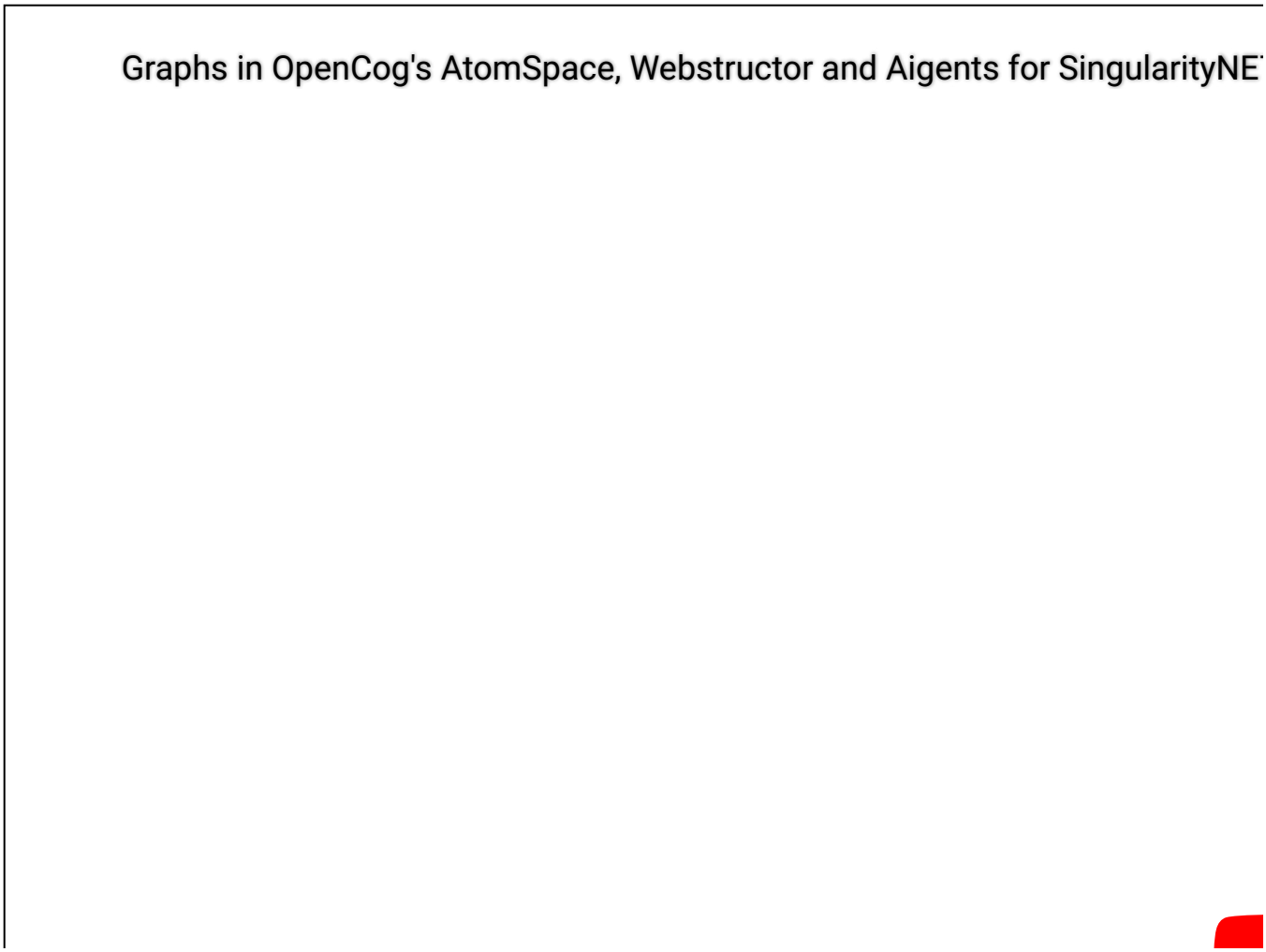
> *Note: We may represent such a situation with more plain binary links between belief owners and atomic elements of their beliefs, but this needs the ability to have links attached to elementary nodes, which is not supported by any graph databases or graph visualization framework in existence, except one: OpenCog, which is an underlying infrastructure component of SingularityNET.*



Example of meta-graph overlaid on plain graph rendered with Aigents Graph — for meta graph to be real, only one link of type **think** is required from Joe to his entire belief sub-graph (Mary likes aerobics, Mary likes Yoga) and so only one link of type **think** is required from node John to his own belief sub-graph (Mary likes hiking, Mary likes yoga).

OpenCog — a **generalized hyper-graph —** is unique simply because it supports all of the above. It does not deal with nodes and links, it deals with **atoms**, where each **atom** is a **generalized hyper-link**, which may have an arity of 0 to represent an atom, or an arity of 2 to represent any plain binary link, or an arity of more than 2 to represent any hyper-link. At the same time, any link, being an atom, may be linked with another link, so the meta-graph can be constructed in several ways.

More introductory information may be found in the following video based on the slides referenced here.

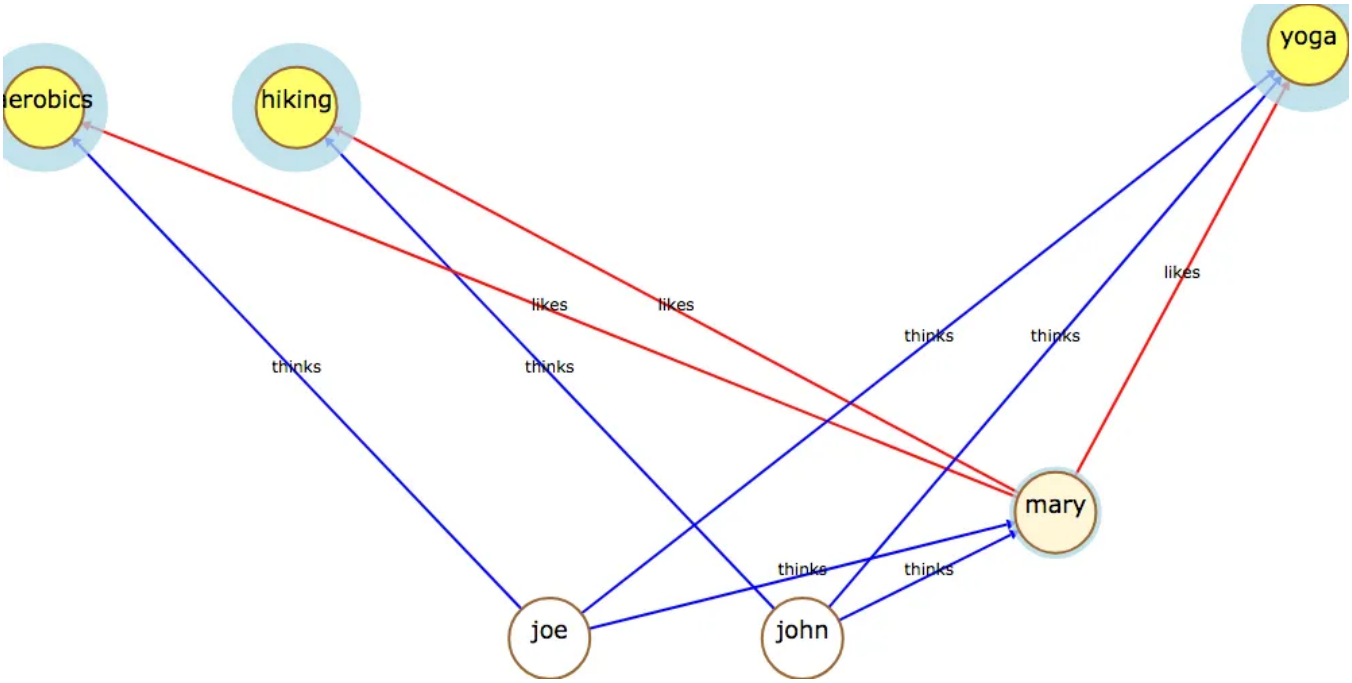


Turning Graphs Into Visual Layouts

Now, let us move from graphs to the visual layouts and features that can be used to represent them.

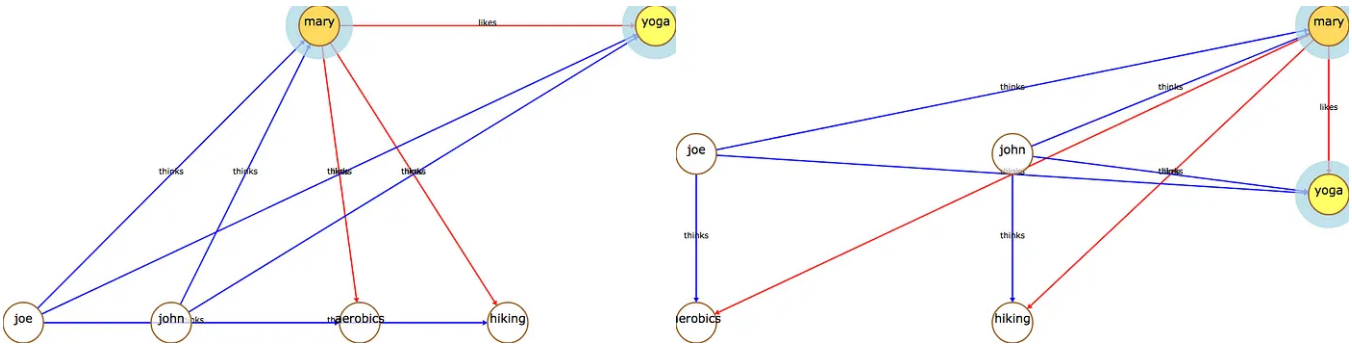
A **layout** is the way nodes and links are arranged on the screen or in the screen windows or on the piece of printed paper. Normally, in most graphical frameworks, layout policy applies to nodes only and the links just follow the arrangement of the links.

Layouts may be **natural**, based on some property of the nodes like name, which can be sorted alphabetically, or by rank or importance of the node. The natural layouts are **deterministic**, as long as the nodes don't have the properties changed. So any time you try to render the same nodes, they will appear in the same way. For instance, deterministic layout in Aigents Graphs is computed with horizontal arrangement performed based on the natural alphabetic order of node name (left to right) and vertical arrangement performed on a basis of node rank (highest — at the top, lowest — at the bottom).



Natural deterministic layout in Aigents Graphs with horizontal arrangement by alphabetic sorting left-to-right by node name and vertical arrangement by node rank with highest on the top and lowest at the bottom.

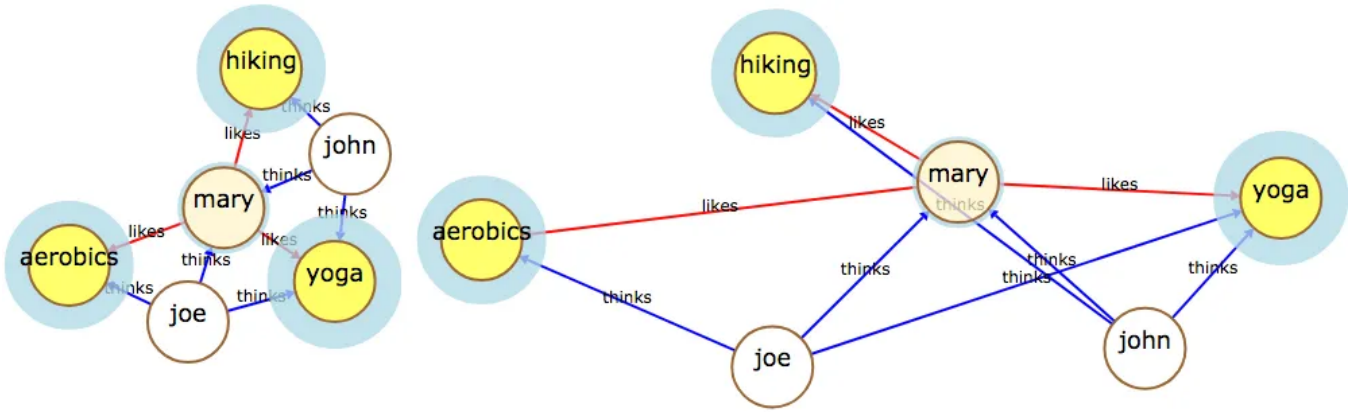
In addition, layouts may be sliced in several layers, so the nodes of different types appear on different layers isolated on graphs spatially in different areas. The **slicing** can be **vertical** (e.g. people on top, activities at the bottom) with horizontal layers, or **horizontal** (e.g. people on the right, activities on the left) with vertical columns.



Horizontal (on the left) and vertical (on the right) slicing rendered with Aigents Graphs

Typically, simulated physical forces are nodes pushing one another in opposite directions, but links are pulling connected nodes together. Optionally, extra forces such as borders or links pushing nodes away may be introduced.

The important tuning property of such **force** layouts is **balance**, which balances the pulling and pushing forces, based on the degree of graph connectivity. This is because, for the same balance value, a loosely connected graph may get spread around the borders with blank space in the middle, while a highly connected graph may appear as an undistinguishable mess of nodes crowded in the middle. So the lower balance may be needed in one case and the higher balance in another.

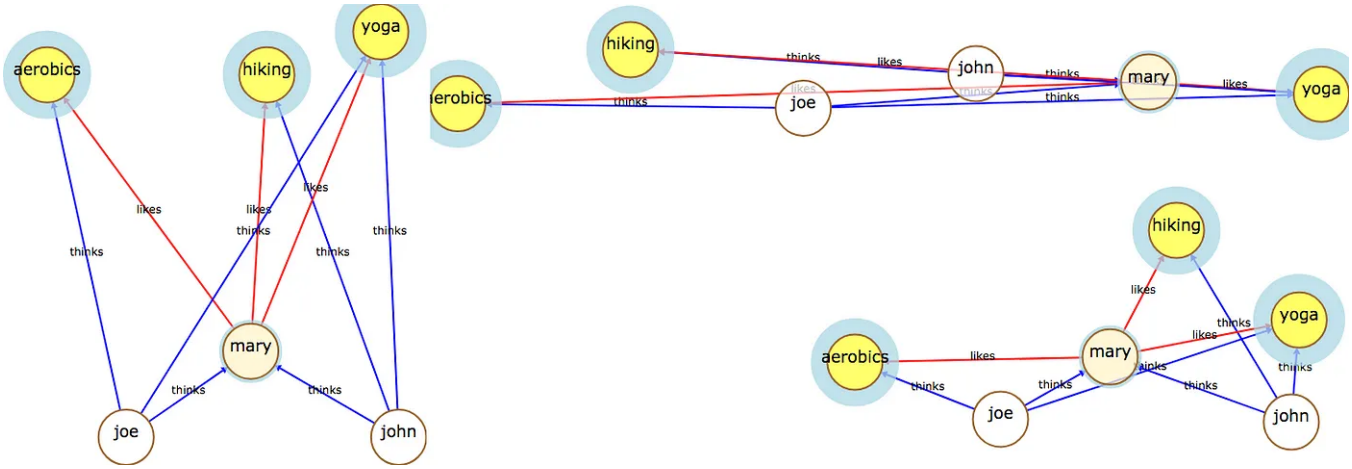


Same graph rendered with low push/pull balance (on the left) and with high push/pull balance (on the right) rendered with Aigents Graphs

The important part of the force layout implementation is the initial arrangement of the nodes on the graph which may be **deterministic** based on some **natural** criteria as discussed above or **random** or **combined**. With complex graphs, a force layout provides a nice looking but barely readable picture, so some extra effort is required to re-arrange some portions of the graph manually in order to get them readable.

The other practical way to reach a reasonable force layout arrangement is to make an initial distribution partially or completely random, and repeat the applying layout under manual control until a reasonable distribution of nodes and links is achieved.

When deterministic initial distribution is involved — and it is used based on some natural order — a force layout may be restricted to limit forces only in a horizontal direction or in a vertical one, or be let to act in both directions.



Force layout with restrictions applied enabling to act in horizontal direction only (on the left), in vertical direction only (on the right-top) and without restrictions to act in both directions (on the right-bottom) as rendered with Aigents Graphs

The other important features of graph visualization include the properties of nodes (vertices) and links (edges) — such as the colours used to render links and nodes interiors and borders, the thickness of node borders, “halos” used to surround the nodes with colours, and the widths of these halos and the widths of the links. This all may be configurable in a way specific to a particular graph used to address a particular problem, and we will have more discussions on these in a future post. Typically, the widths of arrows representing links on weighted graphs correspond to weights or truth values of these links.

Finally, to explain the node colours and “halos” present in all of the examples above, I need to mention that the default saturation and “halo” width in an Aigents Graph corresponds to **node rank**, computed on the basis of **reputation algorithm**. The value of the node rank — depending on the particular graph type — corresponds to the extent the node may play a role in a **root** of a rooted tree graph, or a **hub** in any other graph. Nodes with more link roots or hubs have greater color saturation and greater widths of “halos” around them.

How Can You Get Involved?

Be sure to visit our Community Forum to chat about the research mentioned in the post. Over the coming weeks, we’ll be bringing you more insider access to SingularityNET’s groundbreaking AI research, as well as detailed the specifics of our development. Please refer to our roadmaps for additional information.

SingularityNET Roadmaps

Providing full visibility into our progress and core milestones for SingularityNET’s network architecture.

blog.singularitynet.io

We're proud to work for our incredible community to create a #SingularityForAll

The SingularityNET Team

Big Data | Machine Learning | Artificial Intelligence | Data Science | Research

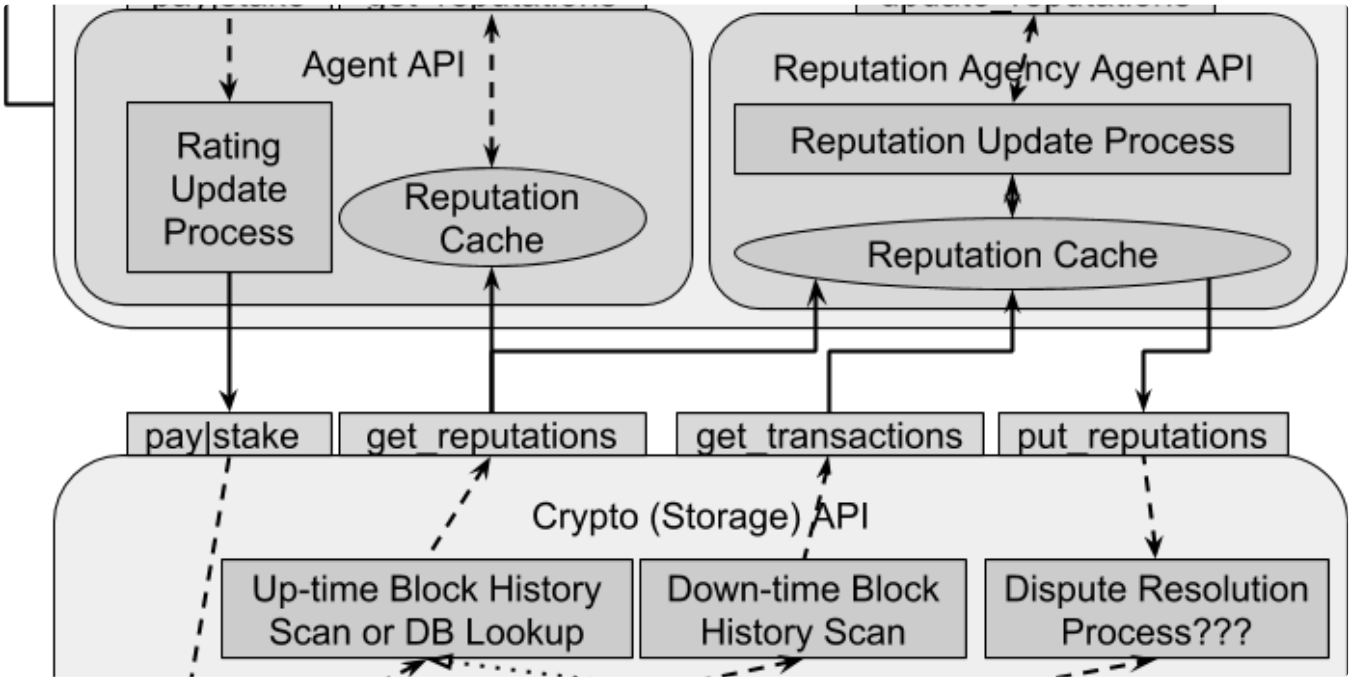
Follow

Written by Aigents with Anton Kolonin

313 Followers · Writer for SingularityNET

Creating personal artificial intelligence and agents of collective intelligence for individuals and small businesses.

More from Aigents with Anton Kolonin and SingularityNET



Aigents with Anton Kolonin in SingularityNET

Reputation System Design for SingularityNET

1. Introduction

19 min read · Apr 12, 2018

1.3K

SingularityNET in SingularityNET

Announcing the Supervisory Council Elections: The Next Step on SingularityNET's Roadmap to...

Dear Singularitarians,

6 min read · Sep 5

176 2

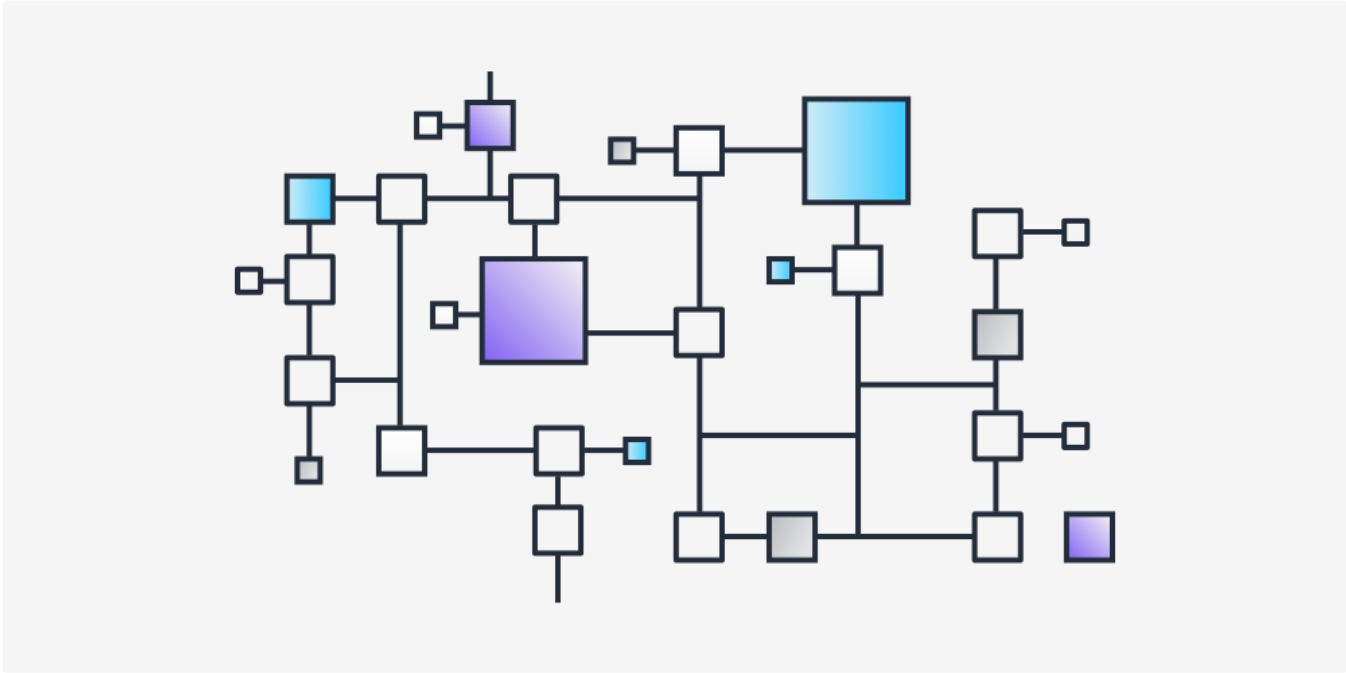
Ben Goertzel in SingularityNET

Decentralized Governance for Decentralized AGI

As Our Progress toward AGI Accelerates, So Must Our Progress on Decentralized Governance

8 min read · Sep 26

155



Aigents with Anton Kolonin in SingularityNET

Integrating OpenCog's Atomspace Hypergraph to Accelerate Intelligence Development

We continue to make progress on platform development.

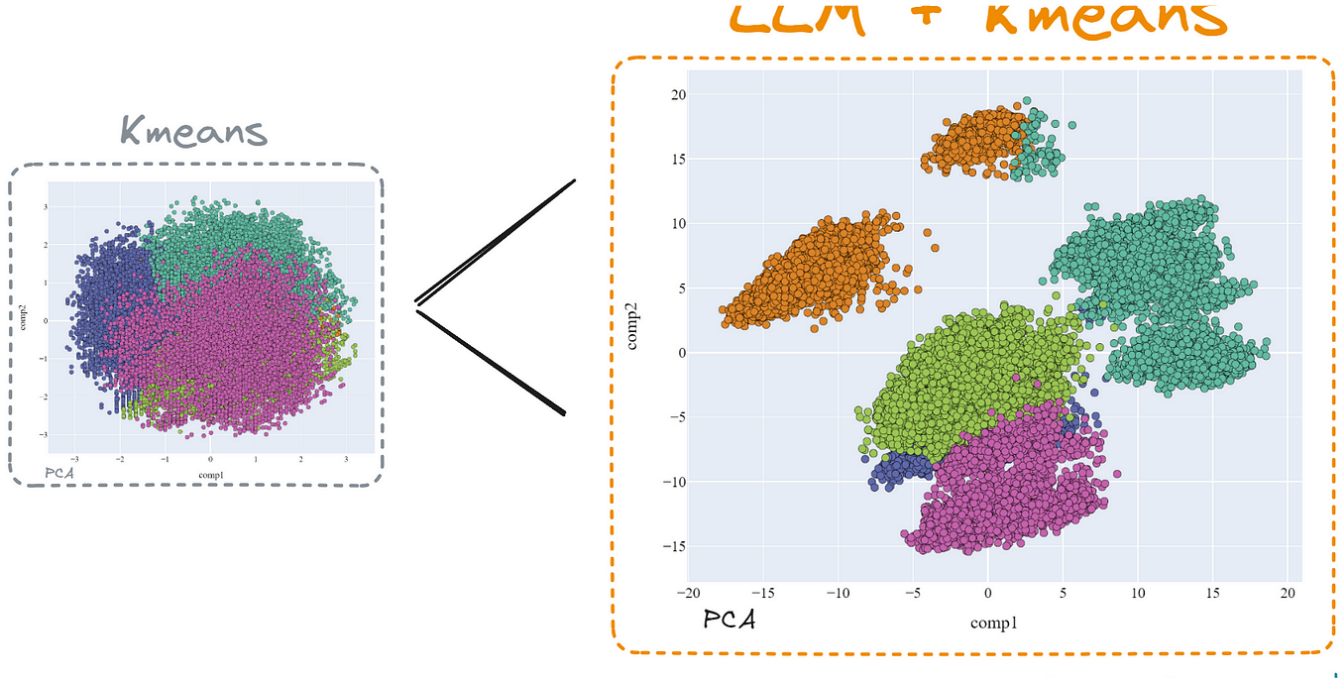
5 min read · Feb 24, 2018

833

See all from Aigents with Anton Kolonin

See all from SingularityNET

Recommended from Medium



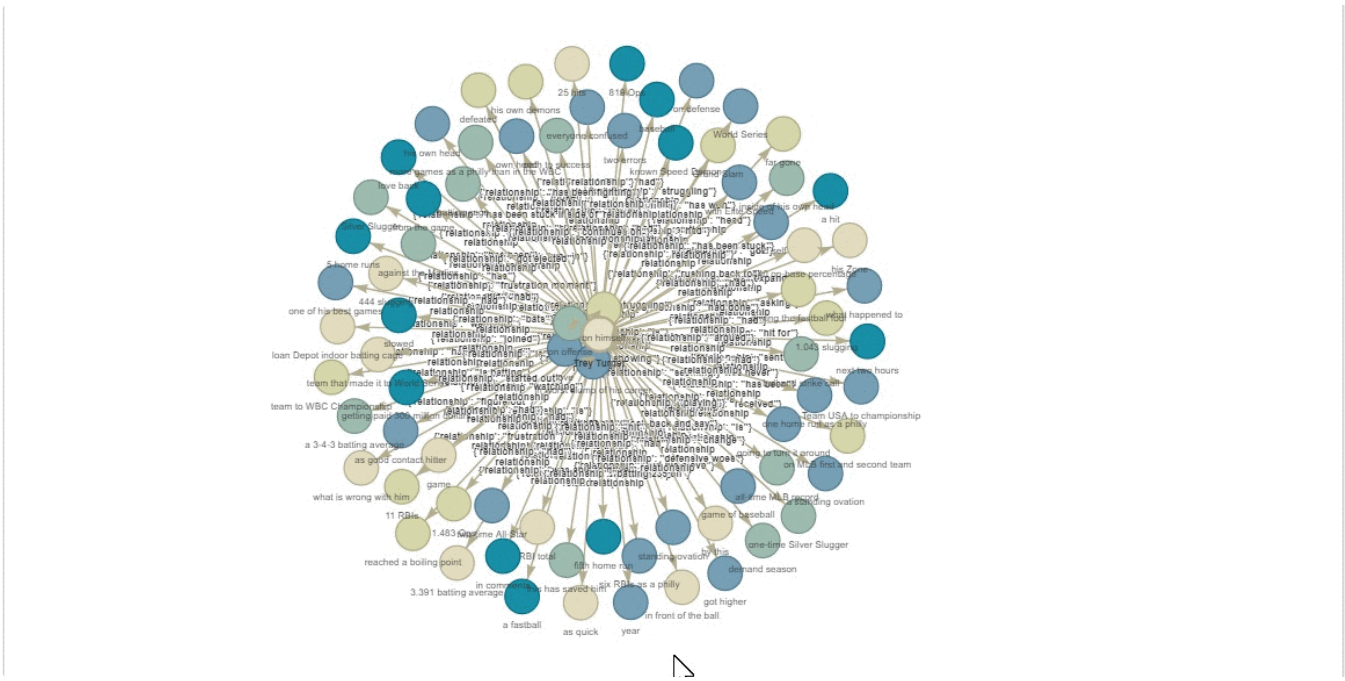
Damian Gil in Towards Data Science

Mastering Customer Segmentation with LLM

Unlock advanced customer segmentation techniques using LLMs, and improve your clustering models with advanced techniques

23 min read · Sep 27

2.4K 22



Wenqi Glantz in Better Programming

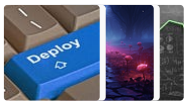
7 Query Strategies for Navigating Knowledge Graphs With LlamaIndex

Exploring NebulaGraph RAG Pipeline with the Philadelphia Phillies

· 17 min read · 5 days ago

590 4

Lists



Predictive Modeling w/ Python

20 stories · 453 saves

Practical Guides to Machine Learning

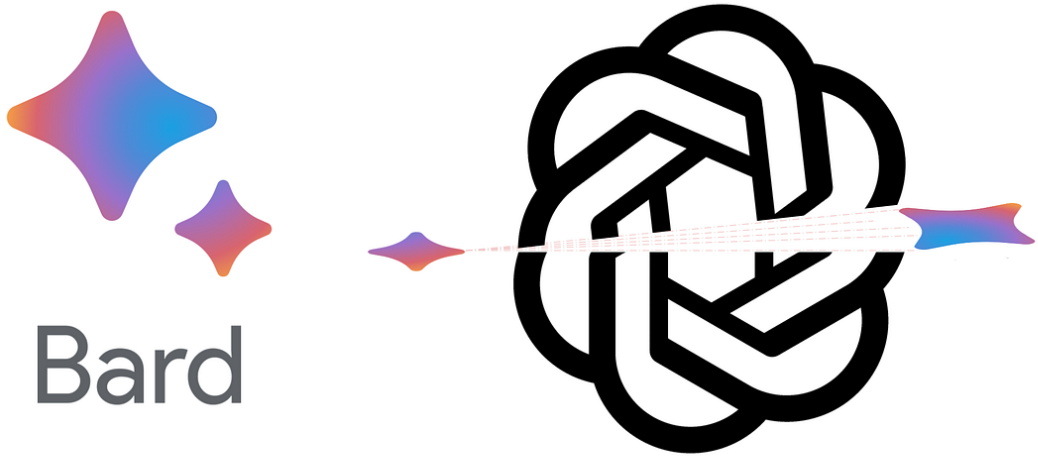
10 stories · 522 saves

Natural Language Processing

669 stories · 285 saves

ChatGPT prompts

24 stories · 466 saves



AL Anany

The ChatGPT Hype Is Over—Now Watch How Google Will Kill ChatGPT.

It never happens instantly. The business game is longer than you know.

· 6 min read · Sep 1

11.6K 361

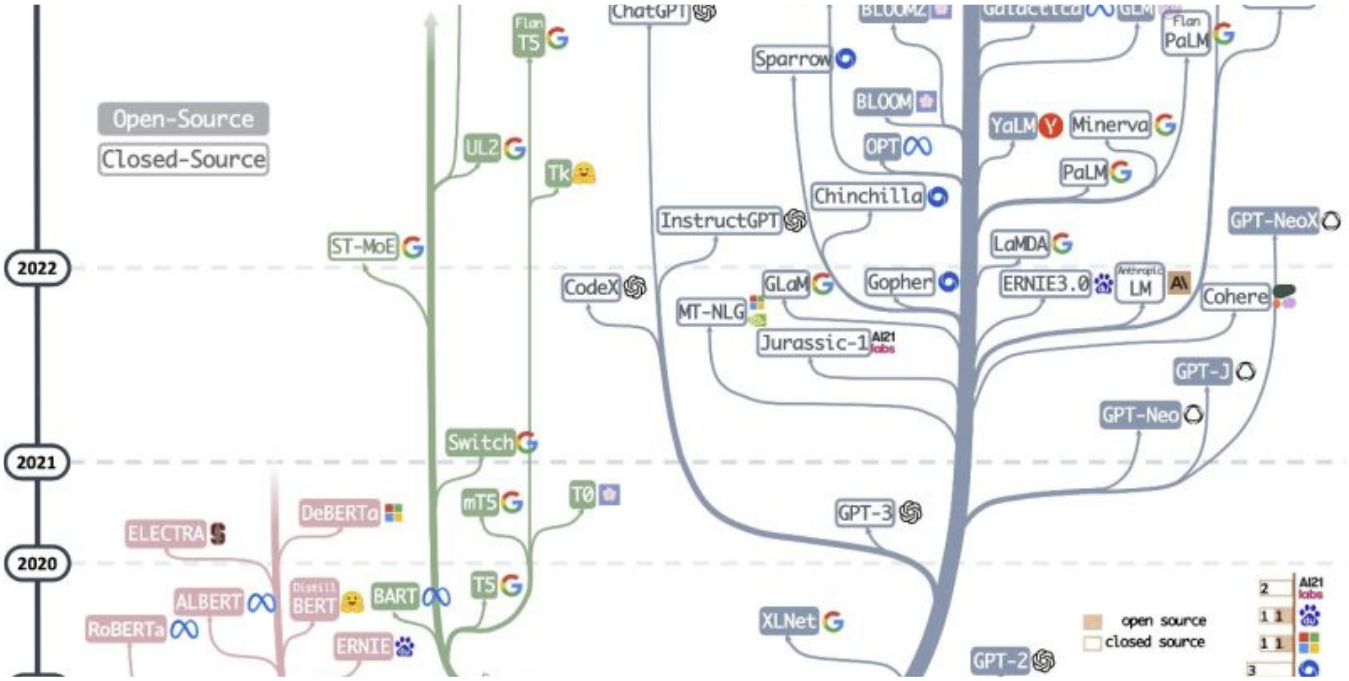
Wes O'Donnell

Could the Army's New Anti-Drone Laser Make a Difference in Ukraine?

If there is one big takeaway from the Ukraine war, it's this: small unmanned aerial systems (sUAS), aka drones, can have an outsized impact...

· 5 min read · 2 days ago

1.4K 19



Haifeng Li

A Tutorial on LLM

Generative artificial intelligence (GenAI), especially ChatGPT, captures everyone's attention. The transformer based large language models...

15 min read · Sep 15

446

Michal Szudejko in The Writing Cooperative

Generating Realistic Synthetic Data with ChatGPT

A Step-by-Step Guide to Understanding and Implementing Synthetic Data Generation Techniques with ChatGPT and Supporting Tools

· 8 min read · 5 days ago

294 3

See more recommendations