# Visualizing an Ontology

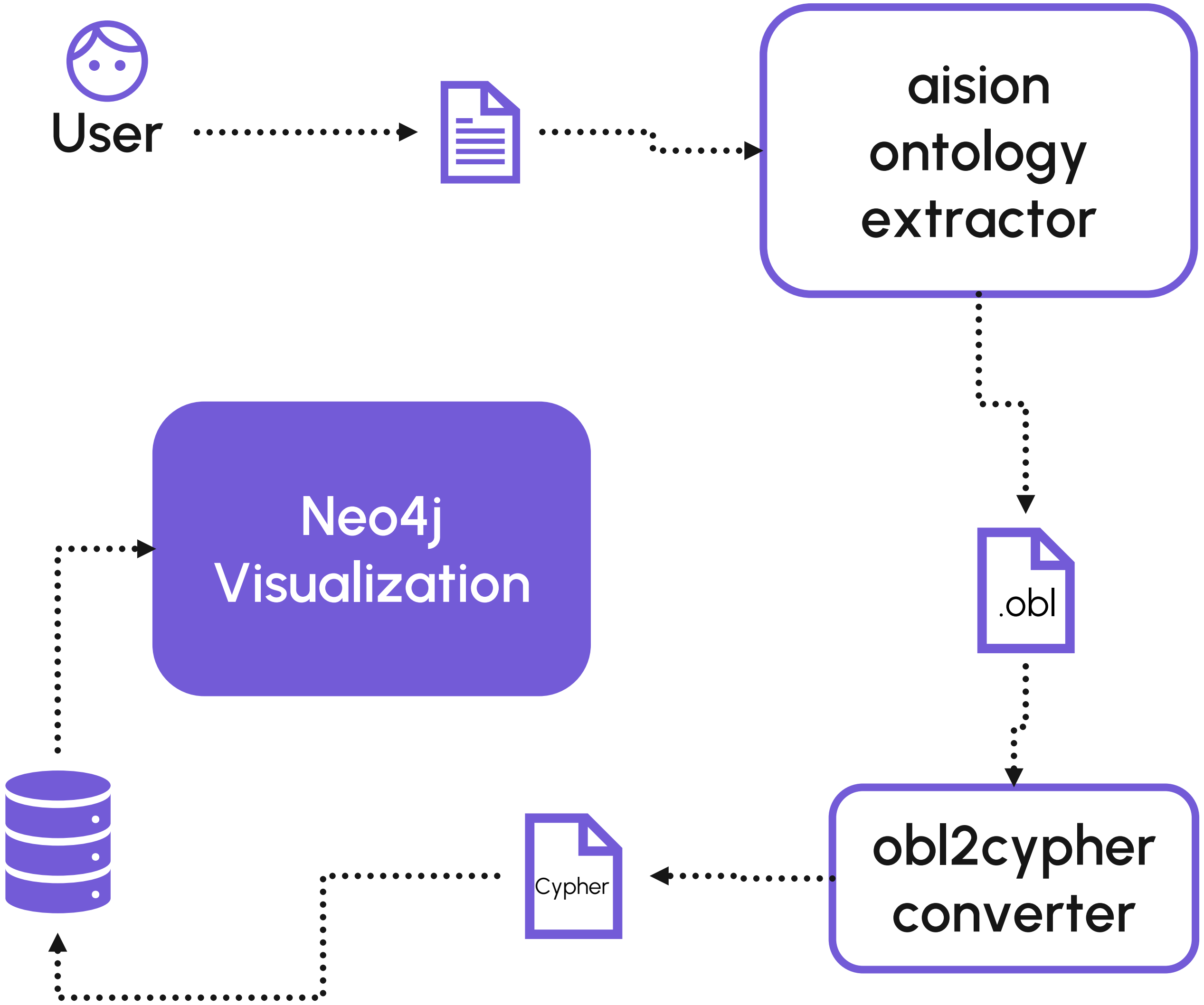
Tolga Keskinoglu

March 9, 2025

aision

# PDF to visualization pipeline

Users upload their PDFs, which are converted to .obl ontology files. The ontology files are converted to Neo4j Cypher which then creates the visualizable graph structure.

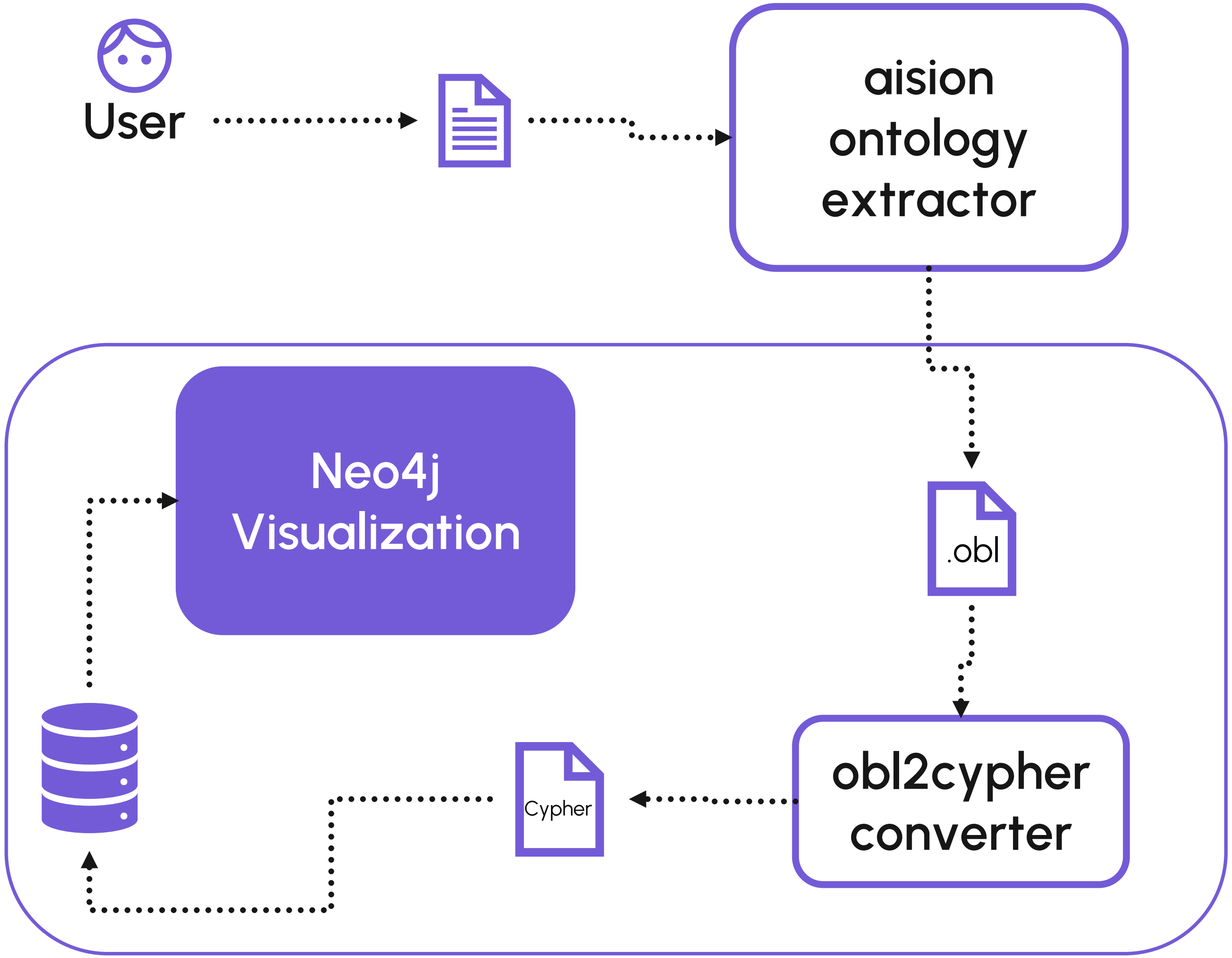



Workflow

aision

# PDF to visualization pipeline

Users upload their PDFs, which are converted to .obl ontology files. The ontology files are converted to Neo4j Cypher which then creates the visualizable graph structure.




Workflow

# Automatic ontology extraction.

Our sample: a 2021 publication about the state of knowledge graphs whose content we use to automatically extract an ontology.

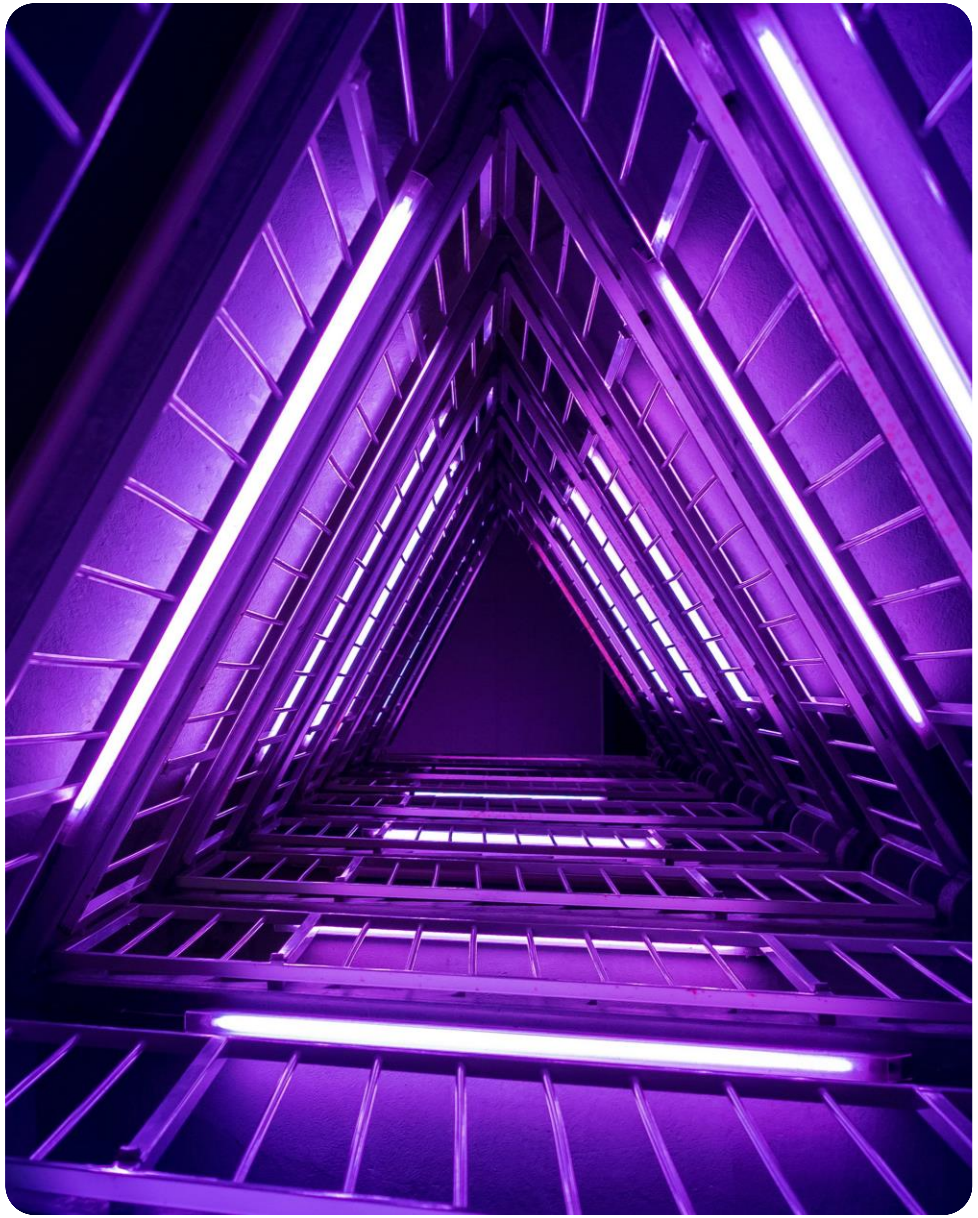
IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, 2021 1

## A Survey on Knowledge Graphs: Representation, Acquisition and Applications

Shaoxiong Ji, Shirui Pan, *Member, IEEE,* Erik Cambria, *Senior Member, IEEE,*
Pekka Marttinen, Philip S. Yu, *Life Fellow, IEEE*

# Anatomy of an ontology.

- Entities
  - Instance : Class [Properties in the form *property* -> *value*].
  - Class :: UpperClass
- Events
  - Instance : Class [Fixed properties such as *time*, *state*, *cause*, *effect*, etc.]
- Relations
  - Instance : Class [*entityA*, *entityB*, attributes -> [*structure*, *function*, *dynamic*, etc.]]



# Entities.

"Knowledge Graph": "Graph Structure" [
  "yearGainedPopularity" -> 2012,
  "description" -> "A graphical representation of knowledge",
  "verificationSimpleOBL" -> [
    "cOBL" -> [
      "value" -> 0.8,
      "thesis" -> "The text describes the historical development of knowledge representation, including the concept of graphical knowledge representation, which is a key aspect of graph structure.",
      "antithesis" -> "However, the term 'Knowledge Graph' specifically refers to a modern concept popularized by Google in 2012, and it may not be directly equivalent to the general notion of graph structure in knowledge representation.",
      "synthesis" -> "While the text does describe the evolution of knowledge representation, including graphical representations, the specific relation between 'Knowledge Graph' and 'Graph Structure' is supported by the historical context and the modern concept of Knowledge Graphs as a type of graph structure. Therefore, the confidence score is 0.8, indicating a strong but not absolute confidence in the correctness of this relation."
    ]
  ]
]
"Graph Structure" :: "Concept".

of about 600 rules. Later, the community of human knowledge representation saw the development of frame-based language, rule-based, and hybrid representations. Approximately at the end of this period, the Cyc project[1] began, aiming at assembling human knowledge. Resource description framework (RDF)[2] and Web Ontology Language (OWL)[3] were released in turn, and became important standards of the Semantic Web[4]. Then, many open knowledge bases or ontologies were published, such as WordNet, DBpedia, YAGO, and Freebase. Stokman and Vries [7] proposed a modern idea of structure knowledge in a graph in 1988. However, it was in 2012 that the concept of knowledge graph gained great popularity since its first launch by Google's search engine[5], where the knowledge fusion framework called Knowledge Vault [3] was proposed to build large-scale knowledge graphs. A brief road map of knowledge base history is illustrated in Fig. 10 in Appendix A. Many general knowledge graph databases and domain-specific knowledge bases have been released to facilitate research. We introduce more general and domain-specific knowledge bases in Appendices F-A1 and F-A2.

# Events.


```
"The concept of knowledge graph gained popularity with its first launch by Google's search engine": "Knowledge Representation Development" [
    "ordinalOBL" -> 9,
    "Time or Moment" -> 2012,
    "State or Condition" -> "Introduction of a new concept in knowledge representation",
    "Cause" -> "Need for a comprehensive and structured knowledge base",
    "Effect" -> "Influence on the development of large-scale knowledge graphs",
    "verificationSimpleOBL" -> [
        "Time or Moment" -> [
            "cOBL" -> [
                "value" -> 1.0,
                "thesis" -> "The text explicitly states that the concept of knowledge graph gained popularity with its first launch by Google's search engine in 2012.",
                "antithesis" -> "There is no counterargument to this statement as it is a direct quote from the text.",
                "synthesis" -> "Based on the information provided, the hypothesis that the attribute Time or Moment has the value '2012' is correct."
            ]
        ],
        "State or Condition" -> [
            "cOBL" -> [
                "value" -> 0.8,
                "thesis" -> "The introduction of the concept of knowledge graph by Google's search engine in 2012 marked a significant milestone in the development of knowledge representation, as it popularized the idea and led to increased adoption and research in the field.",
                "antithesis" -> "The concept of knowledge graph was not entirely new, as it had been proposed earlier by Stokman and Vries in 1988, and other forms of knowledge representation, such as  semantic nets and frame-based languages, had been developed before.",
                "synthesis" -> "While it is true that the concept of knowledge graph was not entirely new, Google's launch of the knowledge graph in 2012 marked a significant turning point in its popularity and widespread adoption, making 'Introduction of a new concept in knowledge representation' a reasonable description of the state or condition associated with this event."
            ]
        ],
        "Cause" -> [
            "cOBL" -> [
                "value" -> 0.8,
                "thesis" -> "The text suggests that the concept of knowledge graph gained popularity due to the need for a comprehensive and structured knowledge base, as evidenced by the development of various knowledge representation systems and standards over the years.",
                "antithesis" -> "However, the text does not explicitly state that the launch of Google's knowledge graph was directly caused by the need for a comprehensive and structured knowledge base. Other factors, such as technological advancements or market demand, might have contributed to its popularity.",
                "synthesis" -> "While the text does not provide direct evidence for the hypothesis, it provides a historical context that suggests the development of knowledge graphs was driven by the need for better knowledge representation and management. The launch of Google's knowledge graph can be seen as a culmination of these efforts, making the hypothesis plausible."
            ]
        ],
        "Effect" -> [
            "cOBL" -> [
                "value" -> 0.8,
                "thesis" -> "The launch of the concept of knowledge graph by Google's search engine in 2012 likely had a significant influence on the development of large-scale knowledge graphs, as it brought attention and resources to the field, leading to further research and innovation.",
                "antithesis" -> "The development of large-scale knowledge graphs may have been an ongoing process, and Google's launch of the concept of knowledge graph in 2012 might not have had a substantial impact on its development, as other factors such as advancements in AI and semantic web technologies could have played a more significant role.",
                "synthesis" -> "While it is possible that the development of large-scale knowledge graphs was already underway, Google's launch of the concept of knowledge graph in 2012 likely accelerated and amplified this process by increasing awareness, investment, and collaboration in the field, ultimately contributing to the growth of large-scale knowledge graphs."
            ]
        ]
    ],
    "verificationDetailedOBL" -> []
].
```


of about 600 rules. Later, the community of human knowledge representation saw the development of frame-based language, rule-based, and hybrid representations. Approximately at the end of this period, the Cyc project[1] began, aiming at assembling human knowledge. Resource description framework (RDF)[2] and Web Ontology Language (OWL)[3] were released in turn, and became important standards of the Semantic Web[4]. Then, many open knowledge bases or ontologies were published, such as WordNet, DBpedia, YAGO, and Freebase. Stokman and Vries [7] proposed a modern idea of structure knowledge in a graph in 1988. However, it was in 2012 that the concept of knowledge graph gained great popularity since its first launch by Google's search engine[5], where the knowledge fusion framework called Knowledge Vault [3] was proposed to build large-scale knowledge graphs. A brief road map of knowledge base history is illustrated in Fig. 10 in Appendix A. Many general knowledge graph databases and domain-specific knowledge bases have been released to facilitate research. We introduce more general and domain-specific knowledge bases in Appendices F-A1 and F-A2.

# Events.

"The concept of knowledge graph gained popularity with its first launch by Google's search engine": "Knowledge Representation Development" [
    "ordinalOBL" -> 9,
    "Time or Moment" -> 2012,
    "State or Condition" -> "Introduction of a new concept in knowledge representation",
    "Cause" -> "Need for a comprehensive and structured knowledge base",
    "Effect" -> "Influence on the development of large-scale knowledge graphs",
    "verificationSimpleOBL" -> [...],
    "verificationDetailedOBL" -> []
].

of about 600 rules. Later, the community of human knowledge representation saw the development of frame-based language, rule-based, and hybrid representations. Approximately at the end of this period, the Cyc project[1] began, aiming at assembling human knowledge. Resource description framework (RDF)[2] and Web Ontology Language (OWL)[3] were released in turn, and became important standards of the Semantic Web[4]. Then, many open knowledge bases or ontologies were published, such as WordNet, DBpedia, YAGO, and Freebase. Stokman and Vries [7] proposed a modern idea of structure knowledge in a graph in 1988. However, it was in 2012 that the concept of knowledge graph gained great popularity since its first launch by Google's search engine[5], where the knowledge fusion framework called Knowledge Vault [3] was proposed to build large-scale knowledge graphs. A brief road map of knowledge base history is illustrated in Fig. 10 in Appendix A. Many general knowledge graph databases and domain-specific knowledge bases have been released to facilitate research. We introduce more general and domain-specific knowledge bases in Appendices F-A1 and F-A2.

# Relations.

```
"The relationship between the knowledge base and its use in knowledge-based systems.":
"Application" [
  "entityAOBL" -> "Knowledge base",
  "entityBOBL" -> "Knowledge-based systems",
  "attributeOBL" -> [
    "Structural" -> "Systematic integration",
    "Functional" -> "Reasoning and problem-solving",
    "Dynamic" -> "Utilization",
    "Abbreviations" -> "None",
    "Citations" -> "None",
    "References" -> "None"
  ],
  "relationCausalOBL" -> [
    "OriginatorOBL" -> "Knowledge base",
    "FunctionalOBL" -> "Knowledge-based systems",
    "ConstraintOBL" -> "The knowledge base is firstly used with knowledge-based systems for
                       reasoning and problem-solving."
  ],
  "verificationSimpleOBL" -> [],
  "verificationDetailedOBL" -> []
].
```

II. OVERVIEW

*A. A Brief History of Knowledge Bases*

Knowledge representation has experienced a long-period history of development in the fields of logic and AI. The idea of graphical knowledge representation firstly dated back to 1956 as the concept of semantic net proposed by Richens [10], while the symbolic logic knowledge can go back to the General Problem Solver [1] in 1959. The knowledge base is firstly used with knowledge-based systems for reasoning and problem-solving. MYCIN [2] is one of the most famous rule-based expert systems for medical diagnosis with a knowledge base

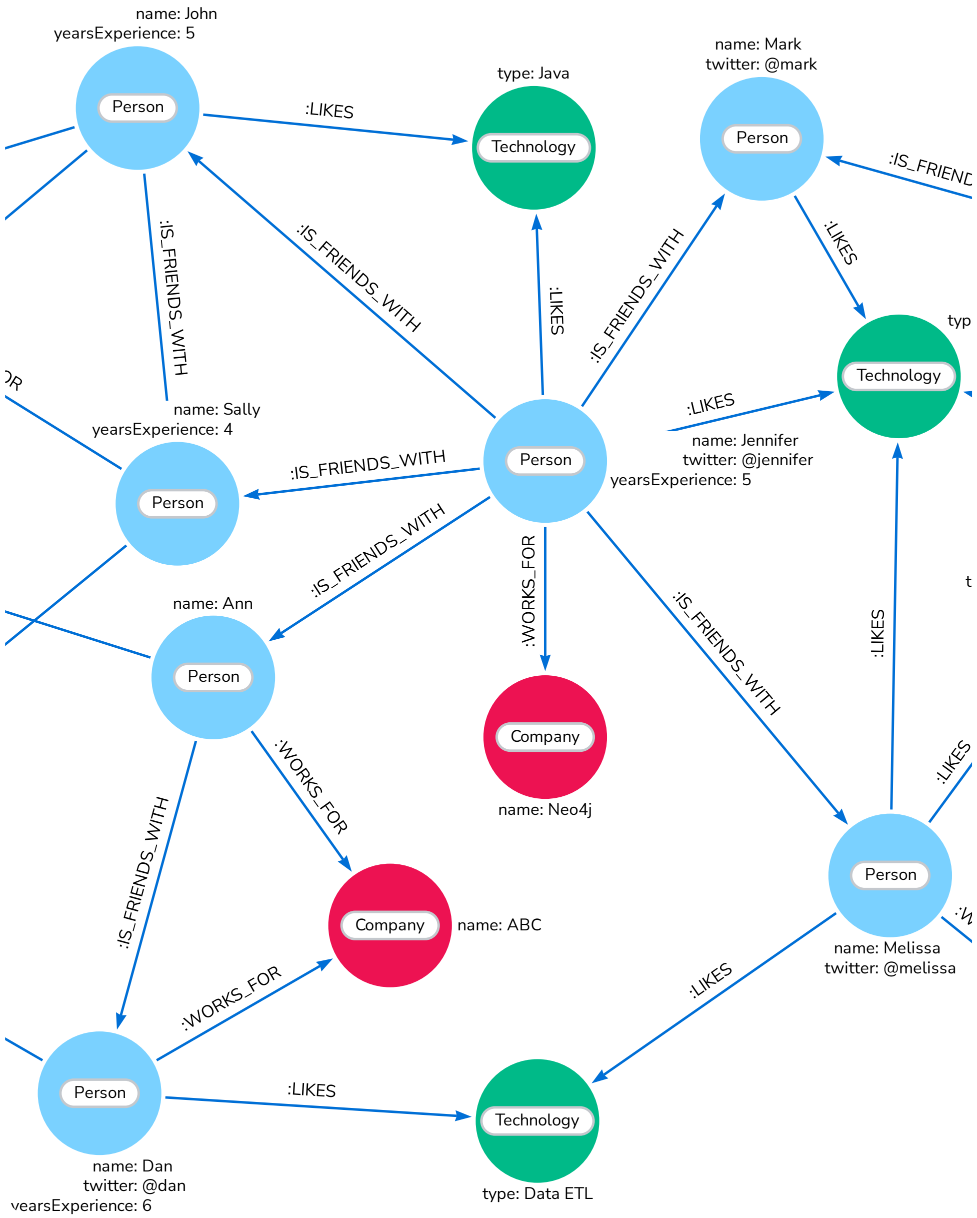
aision

# Neo4j.

Graph database backend that enables storing, modifying, querying, and visualizing graph data.

This enables us to:

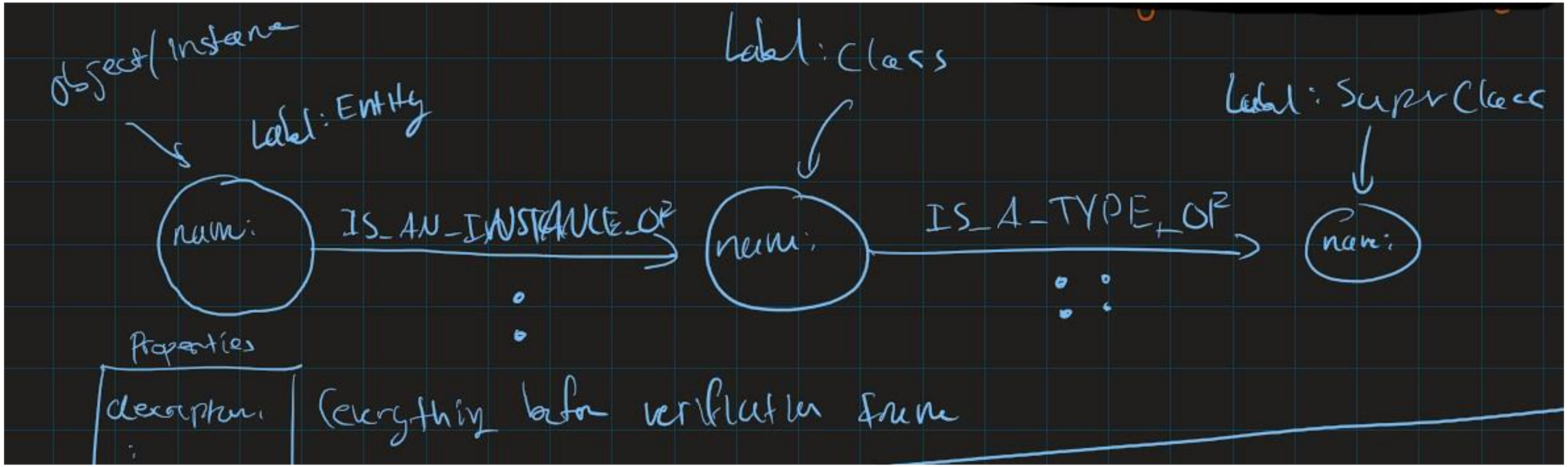
- Visualize ontologies
- Query and manipulate them
- Give industry access to our ontologies in a popular format for linked data structures



aision

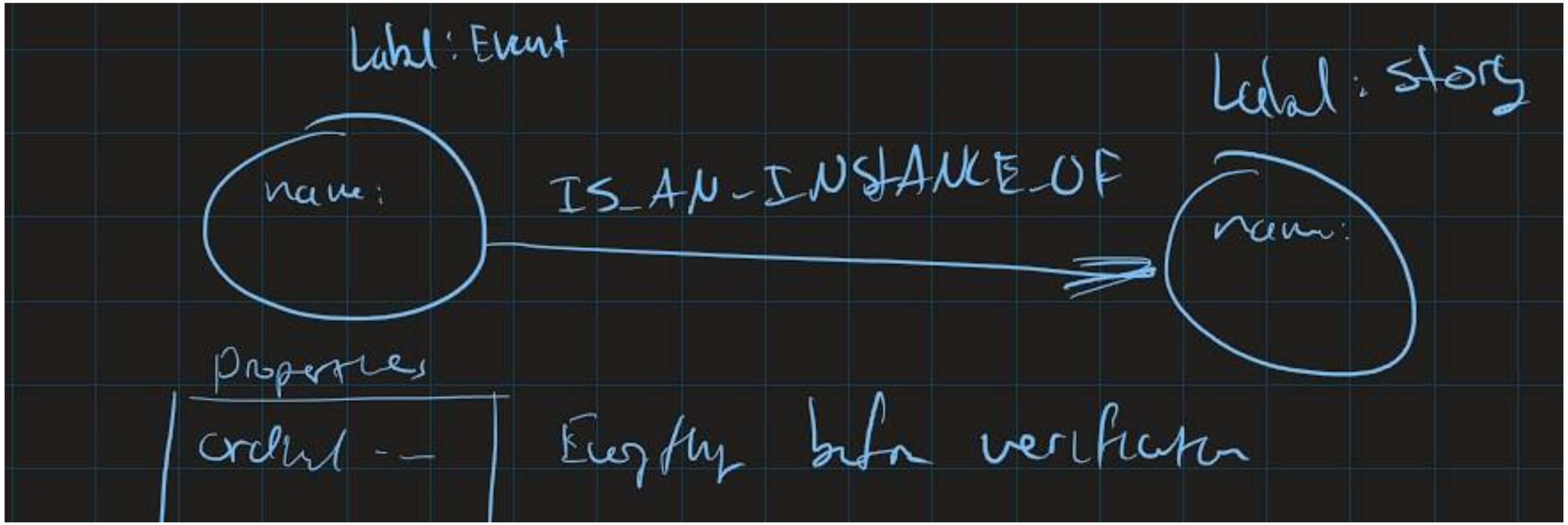
# Entity to Neo4j.

The Entity frame is converted into a 3 node, 2 edge graph where an Object IS_AN_INSTANCE_OF its Class which IS_A_TYPE_OF SuperClass.



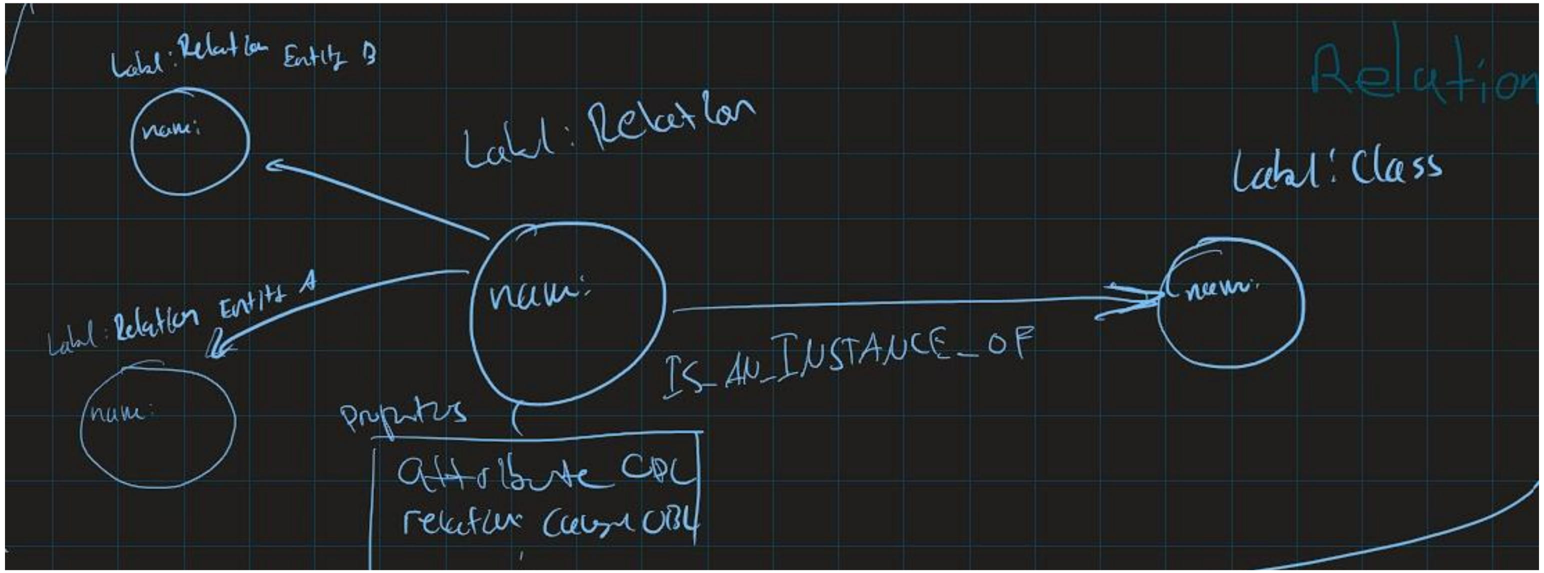
# Event to Neo4j.

The Entity frame is converted into a 2 node, 1 edge graph where an Event IS_AN_INSTANCE_OF its Story (Class). Here there is no super or upper class.



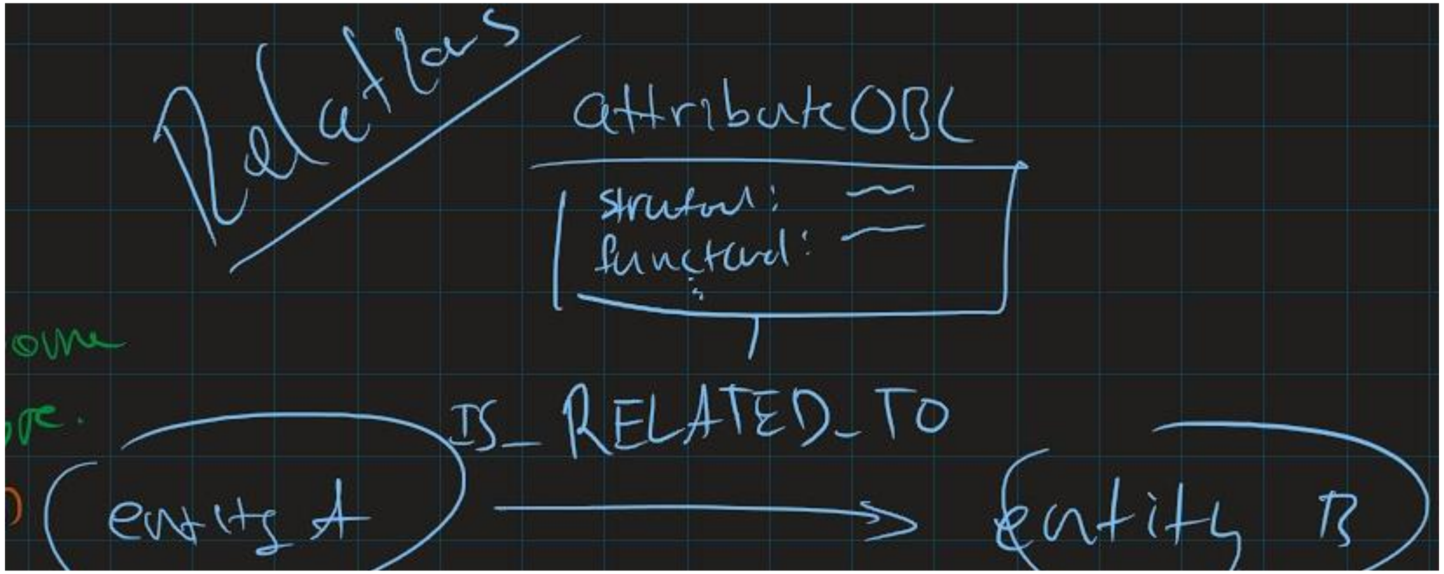
# Relation to Neo4j.

The Relation frame is trickier…



# Relation to Neo4j.

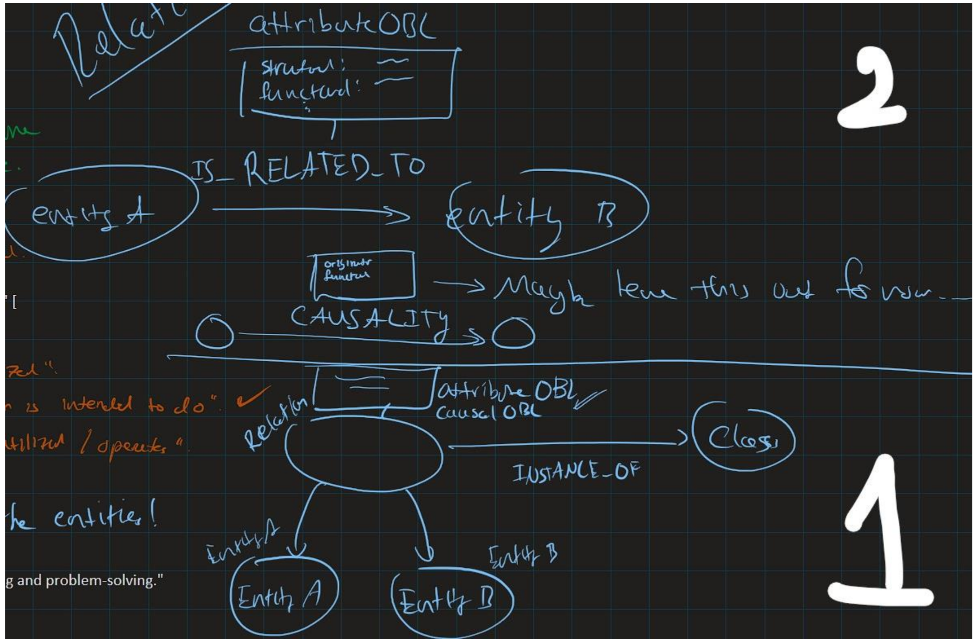
The Relation frame is trickier…



# Relation to Neo4j.

We want entities to be related to one another such that the information in the Relation frame is maximally utilized. However:

1. Entities A and B identified in the Relation frame are not always identified in the Entity frame.
2. The Relation frame is not always internally consistent. The originator of a causal relation may be an entity C which was not identified in either the Relation frame nor in any Entity frame.
3. Neo4j does not support edges to edges (understandably) to preserve the plethora of details each Relation frame contains.



# The obl file.

The .obl file consecutively lists either Entity, Event, or Relation. An entity spans two lines.

```
"researcher"::"human".
"Pekka Marttinen":"researcher"["name"->"Pekka Marttinen", "title"->"None", "affiliation"->"None", "verificationSimpleOBL"->["cOBL"->["value"->0.8, "thesis"->"The input text lists several individuals
"researcher"::"human".
"Philip S. Yu":"researcher"["name"->"Philip S. Yu", "title"->"Life Fellow, IEEE", "affiliation"->"IEEE", "verificationSimpleOBL"->["cOBL"->["value"->0.8, "thesis"->"The input text mentions 'Philip S
"researcher"::"human".
"Affiliation":"Membership"["entityAOBL"->"Shaoxiong Ji", "entityBOBL"->"IEEE", "attributeOBL"->["Structural"->"Author-Organization", "Functional"->"Researcher-Member", "Dynamic"->"None", "Abbreviatio
"Affiliation":"Membership"["entityAOBL"->"Shirui Pan", "entityBOBL"->"IEEE", "attributeOBL"->["Structural"->"Author-Organization", "Functional"->"Researcher-Member", "Dynamic"->"None", "Abbreviations
"Affiliation":"Membership"["entityAOBL"->"Erik Cambria", "entityBOBL"->"IEEE", "attributeOBL"->["Structural"->"Author-Organization", "Functional"->"Researcher-Senior Member", "Dynamic"->"None", "Abb
"Affiliation":"Unknown"["entityAOBL"->"Pekka Marttinen", "entityBOBL"->"None", "attributeOBL"->["Structural"->"Author-Unknown", "Functional"->"Researcher-Unknown", "Dynamic"->"None", "Abbreviations"-
"Affiliation":"Membership"["entityAOBL"->"Philip S. Yu", "entityBOBL"->"IEEE", "attributeOBL"->["Structural"->"Author-Organization", "Functional"->"Researcher-Life Fellow", "Dynamic"->"None", "Abbre
"Human knowledge":"research area"["description"->"formal understanding of the world", "verificationSimpleOBL"->["cOBL"->["value"->0.8, "thesis"->"The text discusses various aspects of knowledge grap
"research area"::"concept".
"Knowledge graphs":"data structure"["purpose"->"represent structural relations between entities", "research_direction"->"cognition and human-level intelligence", "verificationSimpleOBL"->["cOBL"->["
"data structure"::"object".
"Knowledge graph representation learning":"research task"["description"->"learning representations of knowledge graphs", "verificationSimpleOBL"->["cOBL"->["value"->0.9, "thesis"->"The text explicit
"research task"::"activity".
"Knowledge acquisition and completion":"research task"["description"->"acquiring and completing knowledge in knowledge graphs", "verificationSimpleOBL"->["cOBL"->["value"->0.9, "thesis"->"The text e
"research task"::"activity".
"Temporal knowledge graph":"data structure"["description"->"knowledge graph with temporal information", "verificationSimpleOBL"->["cOBL"->["value"->0.8, "thesis"->"The text mentions \"temporal knowl
"data structure"::"object".
```

# Prolog.

Prolog is a logic programming language intended primarily as a declarative programming language: the program is a set of facts and rules, which define relations. A computation is initiated by running a query over the program. [1]

```
mother_child(trude, sally).

father_child(tom, sally).
father_child(tom, erica).
father_child(mike, tom).

sibling(X, Y)      :- parent_child(Z, X), parent_child(Z, Y).

parent_child(X, Y) :- father_child(X, Y).
parent_child(X, Y) :- mother_child(X, Y).
```



[1] https://en.wikipedia.org/wiki/Prolog

# DCG: Definite clause grammar.

DCG notation is just syntactic sugar for normal definite clauses in Prolog which helps with execution efficiency and readability.

```
sentence(A,Z) :- noun_phrase(A,B), verb_phrase(B,Z).
noun_phrase(A,Z) :- det(A,B), noun(B,Z).
verb_phrase(A,Z) :- verb(A,B), noun_phrase(B,Z).
det([the|X], X).
det([a|X], X).
noun([cat|X], X).
noun([bat|X], X).
verb([eats|X], X).
```

```
sentence --> noun_phrase, verb_phrase.
noun_phrase --> det, noun.
verb_phrase --> verb, noun_phrase.
det --> [the].
det --> [a].
noun --> [cat].
noun --> [bat].
verb --> [eats].
```

[1] https://en.wikipedia.org/wiki/Prolog

# OBL grammar in Prolog DCGs.

- "This is a token"
- "object":"class"
- "class"::"superclass"
- "key"->"value", "key2"->"value2"
- [this is a frame]
- Finally, we specify frames into three categories based on their content / structure.

```
token(T) --> "\"", string_without("\"", T), "\"".

object_class(O, C) --> token(O), ":", token(C).

class_superclass(C, S) --> token(C), "::", token(S).

key(K) --> token(K), "->".

value(V) --> "\"", string_with_escaped_quotes(V), "\"".
value(V) --> number(V).

key_value(K,V) --> key(K), value(V).
key_value(K,V) --> key(K), frame(V).

key_value_chain([[Key, Value] | Rest]) -->
    key_value(Key, Value),
    key_value_chain_r(Rest).

key_value_chain_r([[Key, Value] | Rest]) -->
    ", ",
    key_value(Key, Value),
    key_value_chain_r(Rest), ! .
key_value_chain_r([]) --> [] .

frame([]) --> "[]".
frame(F)  --> "[", key_value_chain(F), "]".

obl_frame("Entity", O, C, S, F) -->
    object_class(O, C), frame(F), ".\n" ,
    class_superclass(C, S), ".".
obl_frame("Relation", O, C, [], F) -->
    object_class(O,C), frame(F), { F = [["entityAOBL", _] | _] }, ".".
obl_frame("Event", O, C, [], F) -->
    object_class(O,C), frame(F), { F = [["ordinalOBL", _] | _] }, ".".
```

# Converted Cypher

This is Cypher code which can be used to tell a Neo4j instance the nodes, edges, labels, and properties to create.

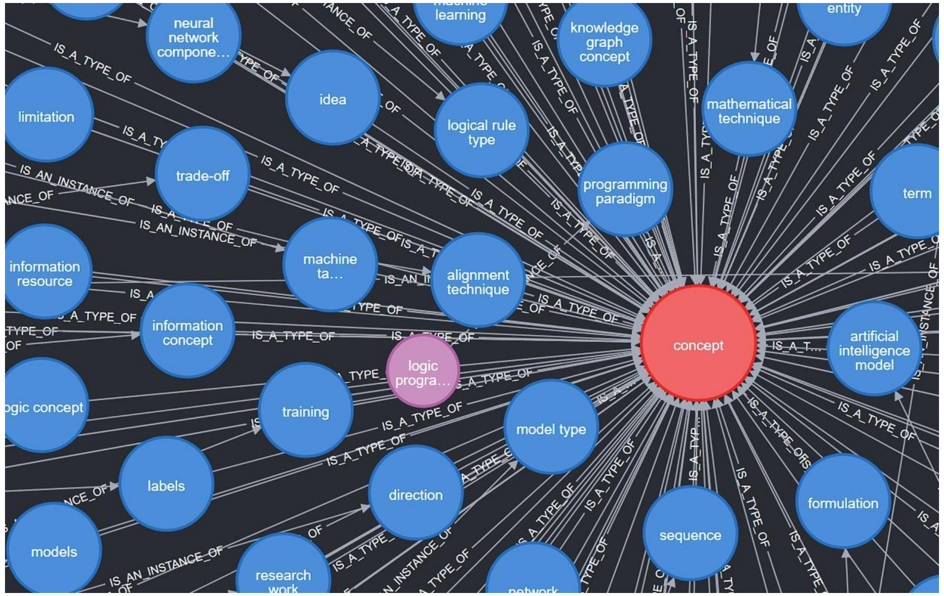
```
  {object: toLower("relevant academic papers"), class: toLower("academic papers"), super: toLower("object")},
  {object: toLower("meaning of query concepts"), class: toLower("query concepts"), super: toLower("concept")},
  {object: toLower("zero-shot image classification"), class: toLower("images"), super: toLower("object")},
  {object: toLower("coherent multi-sentence texts"), class: toLower("texts"), super: toLower("object")},
  {object: toLower("natural language questions"), class: toLower("questions"), super: toLower("object")}
] AS triple
MERGE (o:Entity:EntityObject {name: triple.object})
MERGE (c:Entity:EntityClass {name: triple.class})
MERGE (s:Entity:EntitySuperClass {name: triple.super})
WITH DISTINCT o, c, s
MERGE (o)-[:IS_AN_INSTANCE_OF]->(c)
```

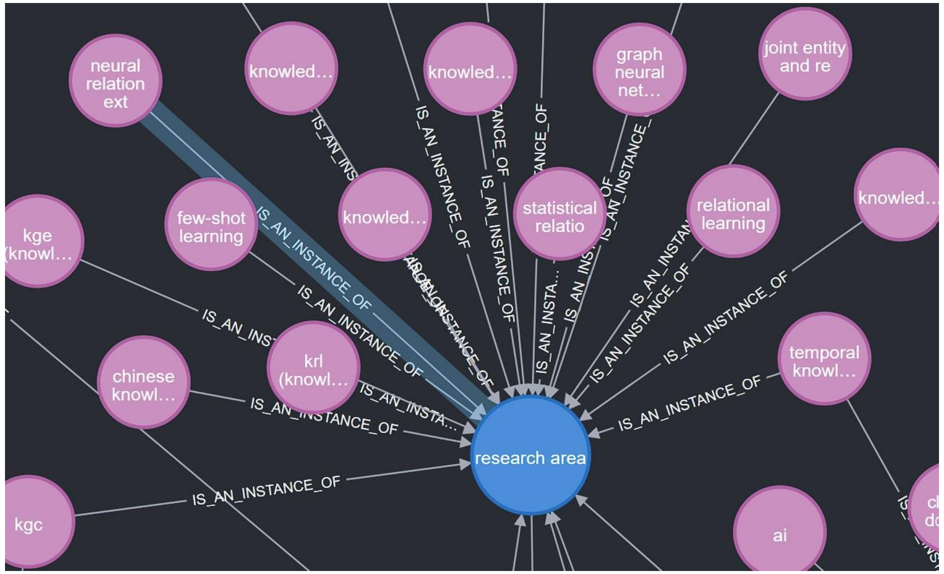
aision

# Entities.

The entities structure is the backbone of the ontology.

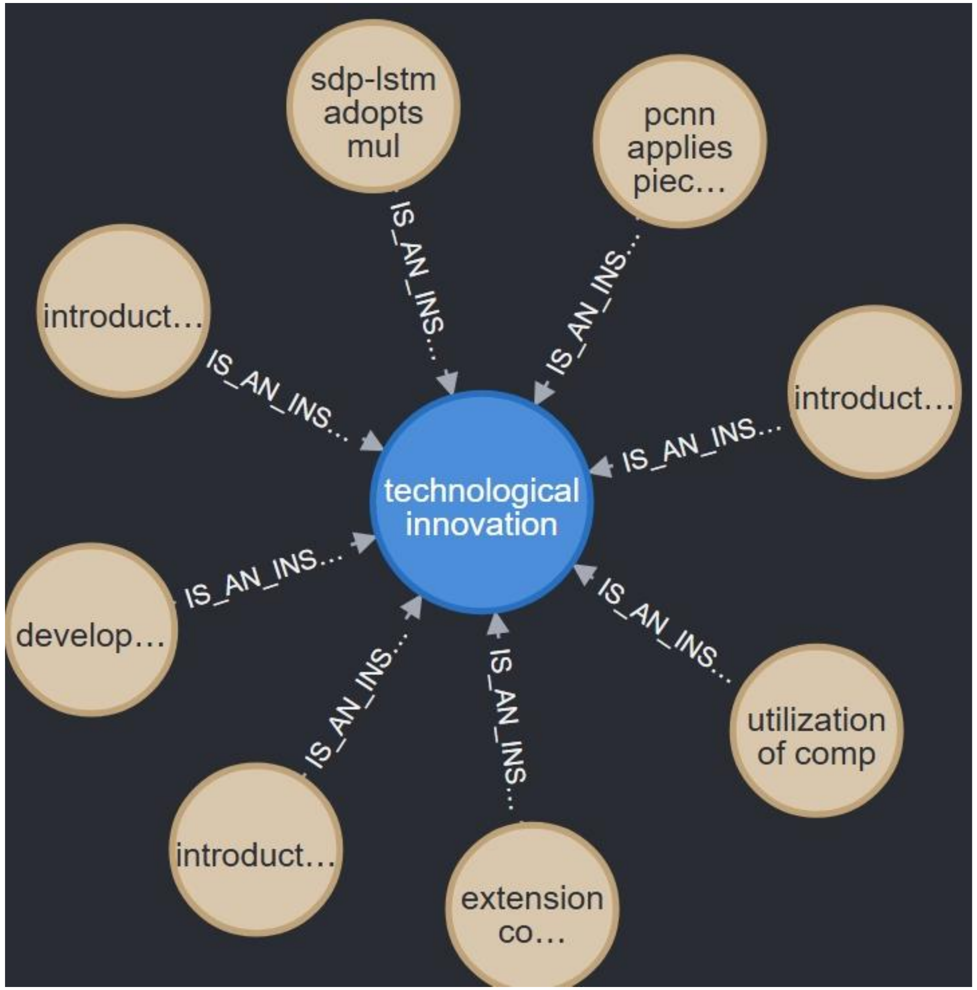
Object/Instances, their Classes and SuperClasses are identified in a hierarchical structure.



# Events.

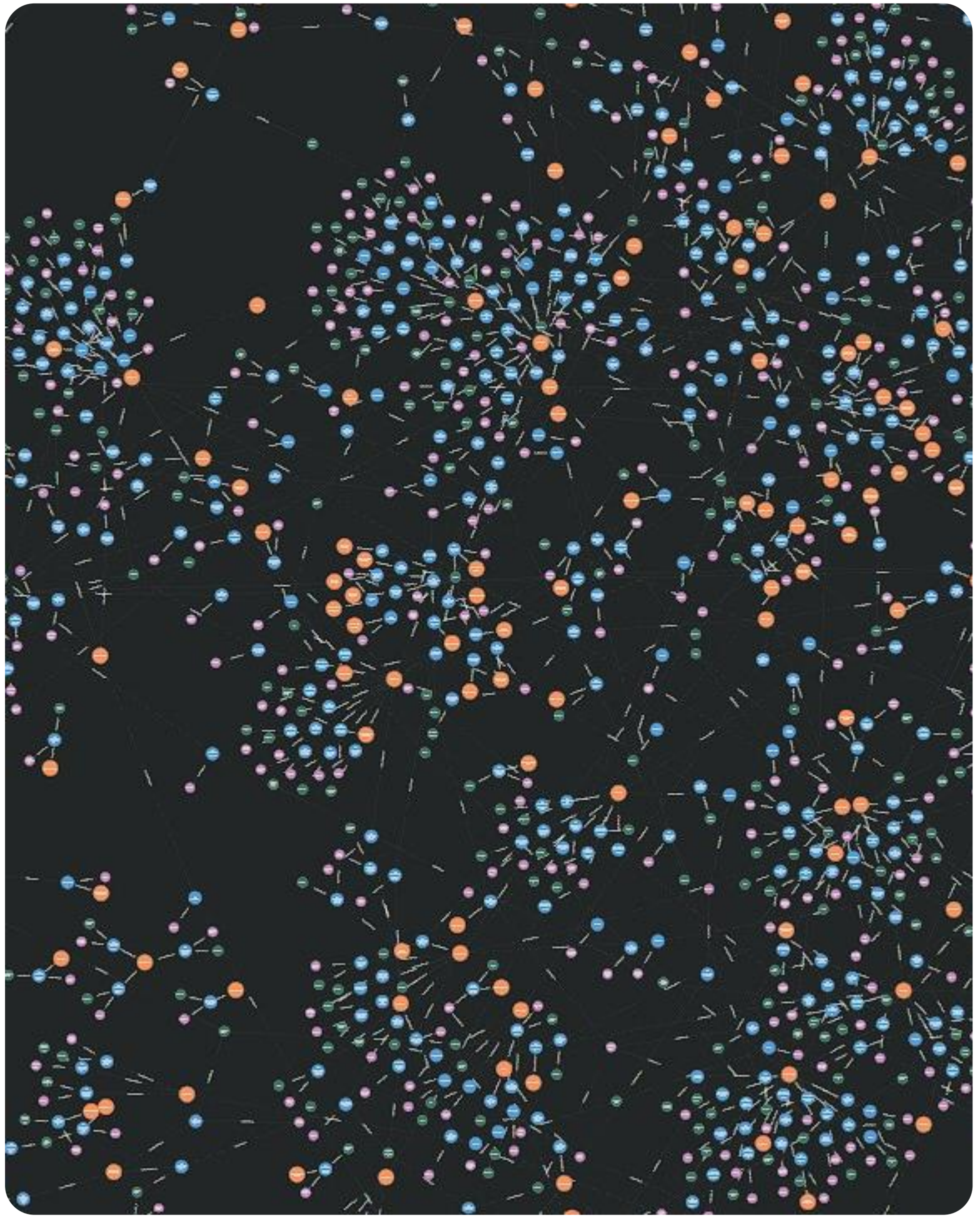
Show individual events that occurred around a larger story.



# Relations

Have the highest interconnectedness of the three frames.

Requires additional effort on how these should be merged into the Entities ontology.



aision

# Performance

The sample paper is 27 pages, from which an obl file with 4,025 lines is created.

The visualization pipeline converts these 4,025 lines into Cypher for all 3 frames (including both versions of the Relation frame) in 92 seconds or about **1.5 minutes** on my AMD Ryzen 7 4800H CPU.

The Cypher code is 3,628 lines long.

```
On the full file.
======================================================================
Total time: 92.922 seconds
======================================================================
Predicate                          Box Entries =    Calls+Redos     Time
======================================================================
string_with_escaped_quotes/3       56,607,098 =  387,716+56,219,382 64.7%
$memberchk/3                       51,309,204 =51,309,204+0          6.3%
dcg_basics:list_string_without/4    2,421,423 =2,421,423+0          6.0%
obl_frame/7                        38,381,915 =38,151,629+230,286    5.1%
parse_obl_relations/2                  71,779 =          1+71,778    3.8%
read_pending_codes/3                    4,674 =     4,674+0          2.2%
\=/2                              112,376,651 =112,376,651+0          2.0%
parse_obl_entities/2                   79,969 =          1+79,968    1.5%
memberchk/2                        51,309,204 =51,309,204+0          1.4%
parse_obl_events/2                     79,417 =          1+79,416    1.3%
token/3                            39,237,703 =39,237,703+0          1.1%
object_class/4                     38,151,629 =38,151,629+0          1.0%
value/3                               824,945 =  774,906+50,039      0.9%
key_value_chain_r/3                   249,423 =  221,969+27,454      0.4%
dcg_basics:string_without/4         2,421,423 =2,421,423+0          0.3%
fill_buffer/1                           4,674 =     4,674+0          0.2%
dcg_basics:number/3                   409,281 =  409,165+116         0.2%
$garbage_collect/1                         67 =        67+0          0.1%
code_type/2                           781,543 =  781,543+0           0.1%
key/3                               1,032,782 =1,032,782+0           0.1%
number_codes/2                        164,429 =  164,429+0           0.1%
key_value/4                           829,532 =  778,318+51,214      0.1%
dcg_basics:digit/3                    781,543 =  781,543+0           0.1%
$attvar:uhook/3                        71,028 =    71,028+0          0.1%
key_value_chain/3                     276,631 =    31,733+244,898    0.1%
```

# Future work.

## Fully automate the pipeline

The current pipelines requires a manual setup of a Neo4j instance with copy and pasting the Cyhper code into the instance. Ideally, a link is presented to a user with a pre-populated Neo4j instance.

## Research on meaningfully increasing connectedness

The Relation frame has a lot of information on how entities should be connected but rules and creative heuristics are needed for when and how these edges should be made.

## Improve performance

The current implementation checks the entire input from each frame down to the end of the file to determine if the current frame is the last frame of its kind. There is likely room for improvement regarding this approach.

# A special thank you to Hermann Rapp and Michael Freund for their support.

March 9, 2025