

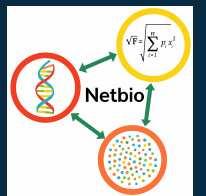
Mini-cours Netbio

Graphes de Connaissances

Construction, visualisation et exploration

Christophe Héliçon, Institut de Génétique et Développement de Rennes, CNRS

Netbio, 15 septembre 2026



Qui suis-je?

Doctorat de Sciences Naturelles à l'ETH Zürich, Suisse

- . Etude de la contribution de la voie de signalisation Wnt/Frizzled au développement du rein embryonnaire chez le Xénope.

Post-Doctorat à l'Université de Lausanne, Suisse

- . Rôle de récepteurs nucléaires dans le syndrome métabolique

Ingénieur de recherche CNRS en expérimentation animale, directeur du CRB Xénopes, CNRS/Univ. Rennes

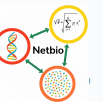
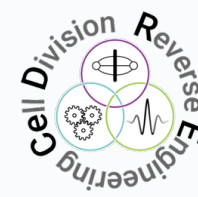
Master en bio-informatique à l'Univ Rennes

Ingénieur de recherche en bioinformatique, IGDR Rennes

Equipe CeDRE: Cell Division Reverse Engineering

Equipe Bingo: Biologie In Slico for (Non-)coding Genome and Oncology

- . Intégration de données hétérogènes et inférence de liens (aide à la décision)



À l'issue de ce cours, vous saurez

1

Définir formellement un graphe de connaissances

et le distinguer d'un graphe ou d'un réseau classique

2

Maîtriser les formalismes de représentation

RDF/OWL, property graphs, ontologies

3

Comprendre la construction, le requêtage et l'exploration

extraction, requêtage, embeddings, GNN

4

Quels outils permettent leur manipulation et visualisation

PARTIE 1 • 10 MIN

Introduction & motivation



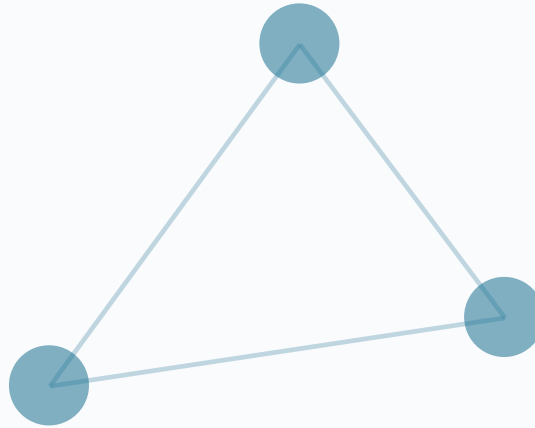
Pourquoi représenter la connaissance sous forme de graphe ?



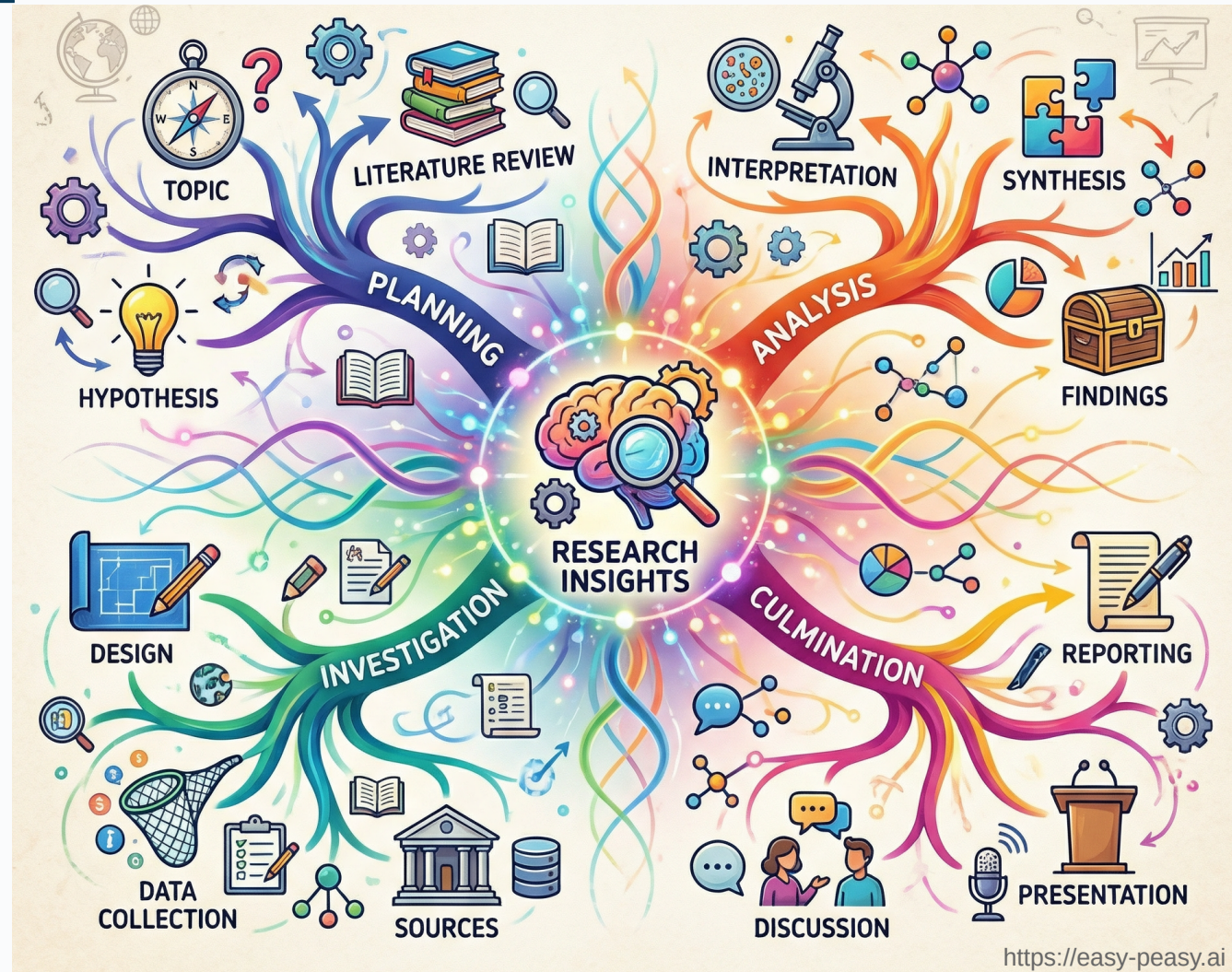
Un graphe



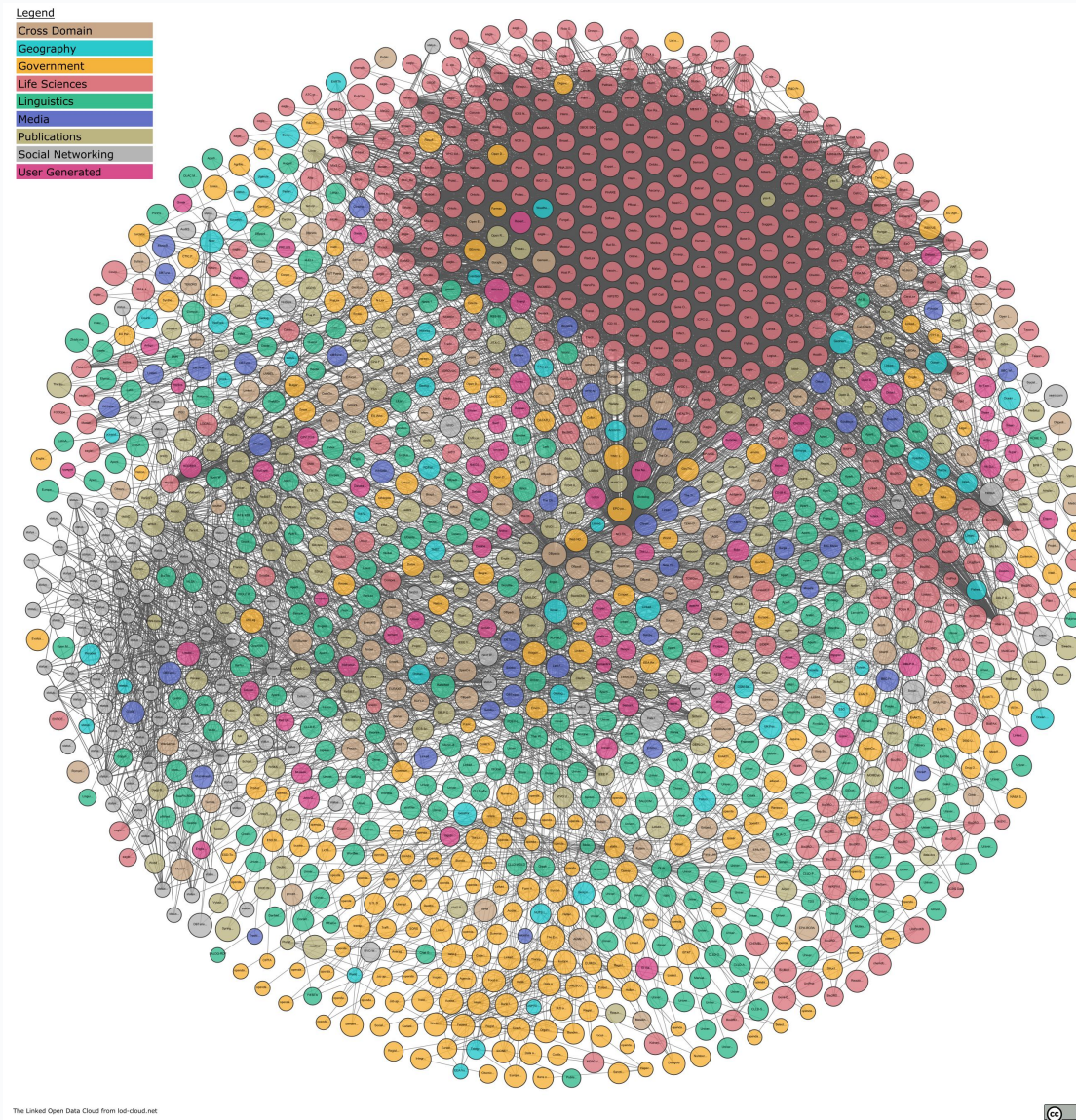
Un autre graphe



Un autre graphe



Un graphe de graphes



Pourquoi utiliser des graphes?

Organiser/structurer

Réunir

Visualiser

Fouiller

Trouver

Ce qu'apporte un graphe

- Un représentation “plus réelle” du monde
- Un monde hétérogène = un graphe hétérogène
- Un monde multiple
- Un monde ouvert
- Une représentation comprise par l’humain et la machine

Pourquoi structurer la connaissance en graphe(s)?

Le monde réel n'est pas une table

- Relations nombreuses, hétérogènes et souvent n-aires
- Schéma évolutif : de nouveaux types de relations apparaissent en continu
- Jointures multiples → requêtes lourdes, perte de sens
- Difficile d'intégrer des sources externes hétérogènes

Ce qu'apporte un graphe de connaissances

- Un schéma flexible, centré sur les entités et leurs relations
- Une information stockée directement (relations entre entités)
- Une sémantique explicite (types, ontologies)
- Une intégration naturelle de sources hétérogènes
- Une exploration native (communautés, plus court chemin...)
- Une inférence automatique (transitivité, hiérarchie)
- Un support pour le raisonnement et l'apprentissage
- Une traçabilité (FAIR)

Comment choisir la représentation de mes données?

Qu'est-ce qui est important dans mes données?

Les entités → tables, SQL

Les relations entre entités → graphe

PARTIE 2 • 10 MIN

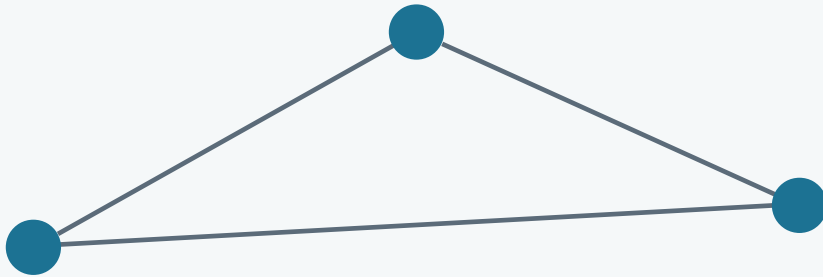
Les types de graphes

Une famille d'objets aux propriétés combinables.



Graphe simple et graphe dirigé

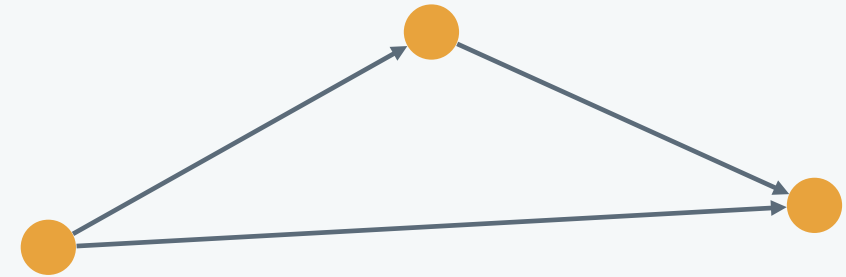
Graphe simple



Noeuds reliés par des arêtes

Exemple : un réseau d'interactions protéine-protéine (PPI) brut : « A interagit avec B »

Graphe dirigé

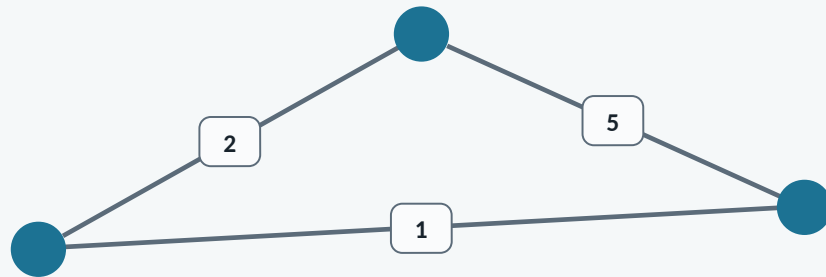


Les arêtes ont un sens : d'un noeud source vers un noeud cible.

Exemples : un réseau de régulation génique (facteur de transcription → gène cible), une voie de signalisation, un réseau trophique (proie → prédateur).

Graphe pondéré et graphe dirigé pondéré

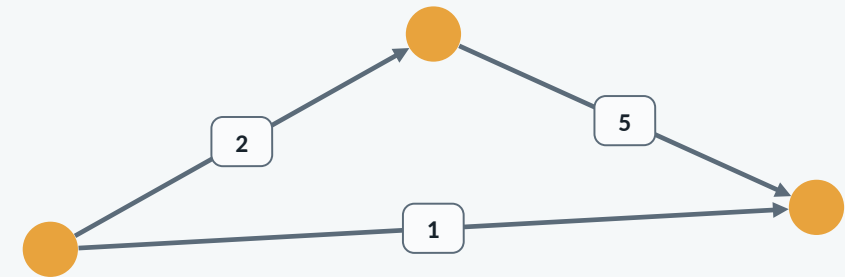
Graphe pondéré



Chaque arête porte une valeur numérique quantifiant la force ou le coût de la relation.

Exemples: un réseau de co-expression (*poids* = corrélation), un réseau PPI type STRING (*poids* = score de confiance).

Graphe dirigé pondéré



Combine sens et intensité : direction du lien et force associée.

Exemple : un réseau de flux métaboliques (*direction* = sens de la réaction, *poids* = flux mesuré)

Multigraphe et graphe biparti

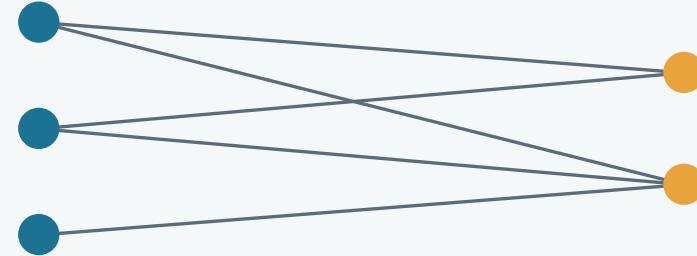
Multigraphe



Plusieurs arêtes (souvent de nature différente) peuvent relier la même paire de noeuds.

Exemple : deux protéines reliées à la fois par une interaction physique, une interaction génétique et une co-expression

Graphe biparti

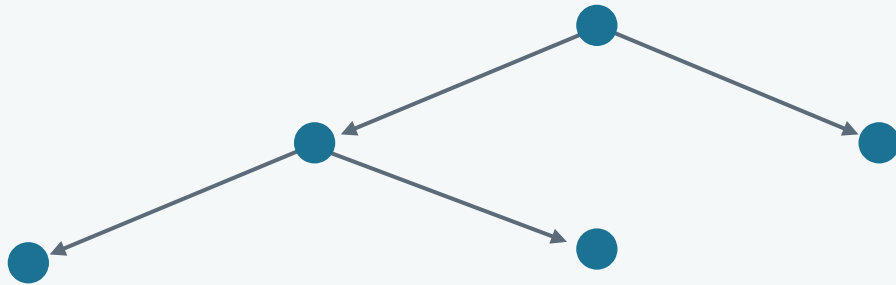


Deux ensembles de noeuds disjoints ; les arêtes ne relient que d'un ensemble à l'autre.

Exemples : réseau gène-maladie (DisGeNET), réseau médicament-cible, réseau hôte-pathogène.

Arbre, graphe orienté acyclique

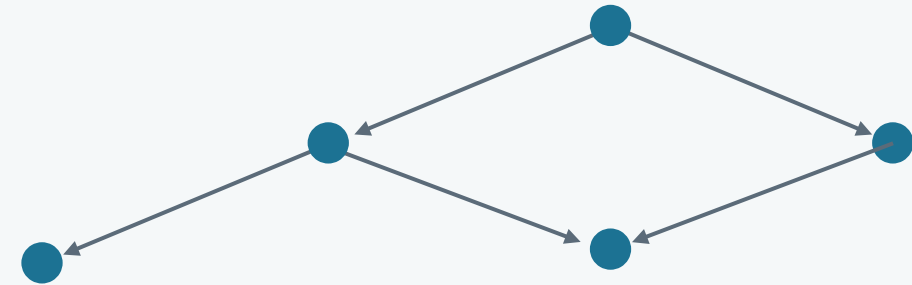
Arbre



Arbre : connexe et acyclique, chemin unique entre deux noeuds.

Exemple : un arbre phylogénétique (arbre).

DAG

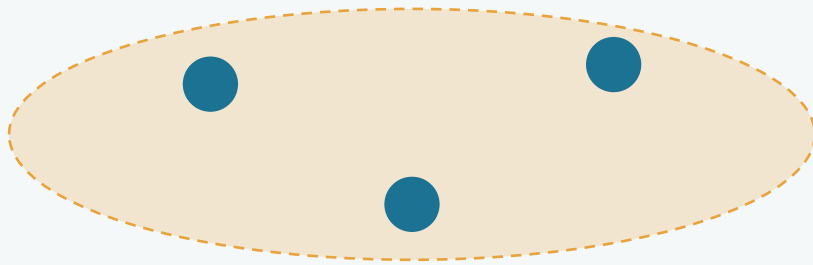


DAG : dirigé et acyclique, un noeud peut avoir plusieurs parents.

Exemple : la Gene Ontology, où un terme peut avoir plusieurs parents (DAG, pas un arbre strict).

Arbre, graphe orienté acyclique et hypergraphe

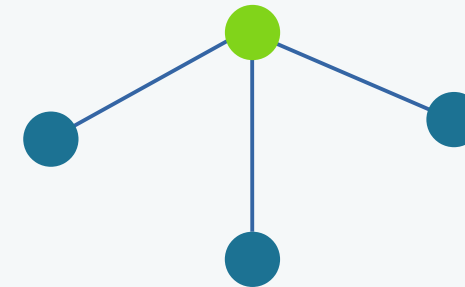
Hypergraphe



Une hyper-arête peut relier plus de deux noeuds à la fois — au-delà des relations binaires.

Exemple: une réaction enzymatique (substrat + enzyme + cofacteur + produit) ; les interactions protéiques d'ordre supérieur.

Réification

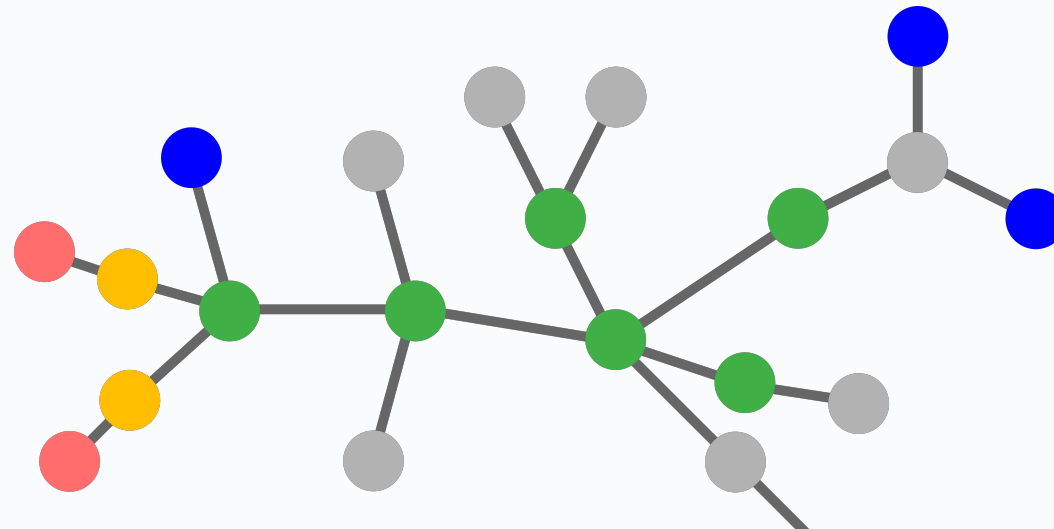


Un nouveau noeud est créé pour représenter cette hyper-relation.

Exemple : une réaction enzymatique (substrat + enzyme + cofacteur + produit) ; les interactions protéiques d'ordre supérieur.

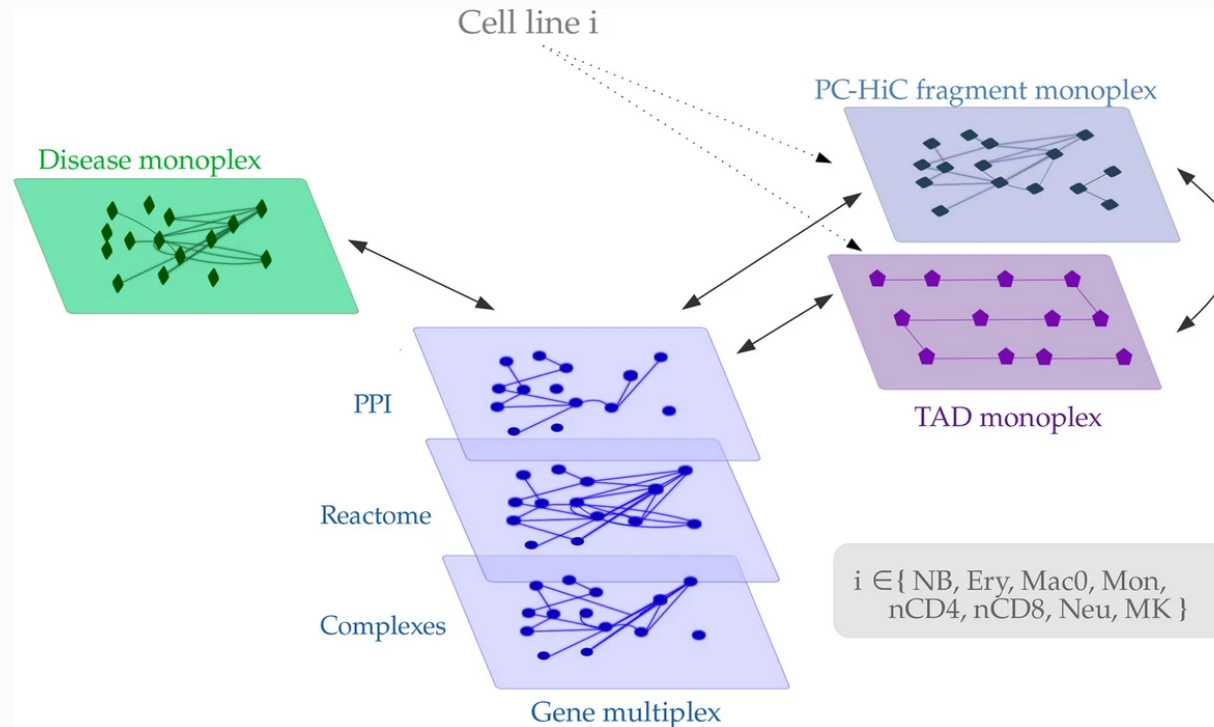
Graphe hétérogène

- Plusieurs types de noeuds et/ou de relations coexistent dans la même structure
- C'est le premier pas vers le graphe de connaissances : on quitte un objet mathématique unique pour représenter un système réel

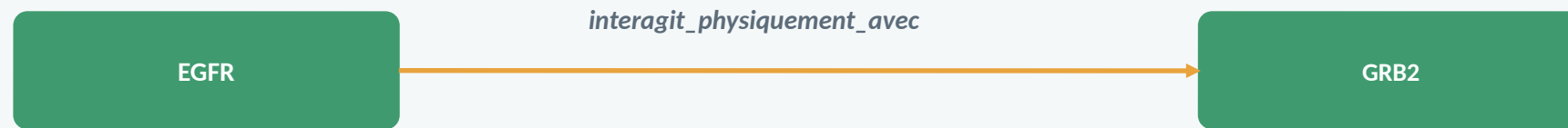


Réseau multiplex / multicouche

- Les mêmes noeuds sont connectés à travers plusieurs « couches », chacune représentant un type de relation différent
- Permet de combiner des sources hétérogènes sans les fondre en un seul graphe indifférencié



Graphe de connaissances

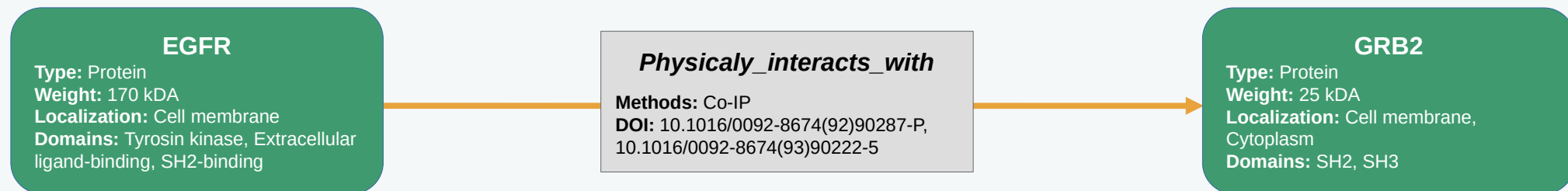


Entités et relations typées, organisées autour d'un schéma/ontologie explicite, support du raisonnement.

Exemples : *Hetionet*, *AgroLD*

Graphe de propriétés

Graphe de propriétés



Noeuds et arêtes portent des attributs libres (clé-valeur), sans schéma formel type OWL.

Exemple: PrimeKG ou OpimusKG

PARTIE 3 • 10 MIN

Qu'est-ce qu'un graphe de connaissances ?

Une définition formelle, pour poser un vocabulaire commun avant d'aller plus loin.

Définition formelle

$$G = (E, R, L, T)$$

entités **E** · relations **R** · littéraux **L** · triplets $T \subseteq E \times R \times (E \cup L)$

Chaque fait du graphe s'exprime comme un triplet :



Un graphe de connaissances = un ensemble de triplets, typés et reliés par un schéma (ontologie).

sujet + prédicat + objet $\Leftrightarrow$ head + relation + tail

Graphe simple, réseau, graphe de connaissances



Graphe simple

- Noeuds + arêtes
- Pas de sémantique
- Un seul type de relation



Réseau (biologique...)

- Noeuds homogènes
- Arêtes pondérées
- Ex. réseau PPI, réseau de gènes



Graphe de connaissances

- Entités et relations typées
- Schéma / ontologie explicite
- Support du raisonnement

Les composants d'un graphe de connaissances

E

Entités

Les « choses » du monde : un gène, une protéine, une maladie, un médicament...

R

Relations

Les prédicats qui relient deux entités : code_pour, interagit_avec, traite...

L

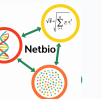
Littéraux

Des valeurs concrètes attachées à une entité : un identifiant, une masse moléculaire, une date

T

Types / classes

La catégorie d'une entité, définie par le schéma (rdfs) ou l'ontologie (owl) : Gène, Protéine, Maladie

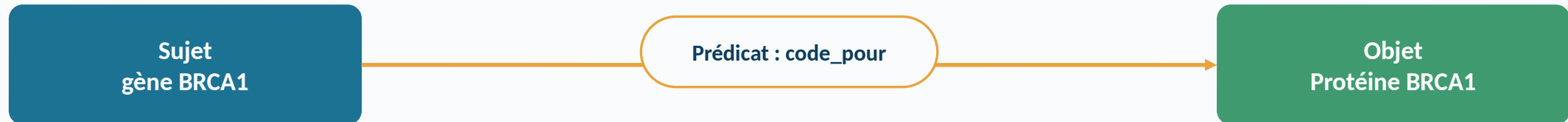


Définition formelle

$$G = (E, R, L, T)$$

entités **E** · relations **R** · littéraux **L** · triplets $T \subseteq E \times R \times (E \cup L)$

Chaque fait du graphe s'exprime comme un triplet :



Un graphe de connaissances = un ensemble de triplets, typés et reliés par un schéma (ontologie).

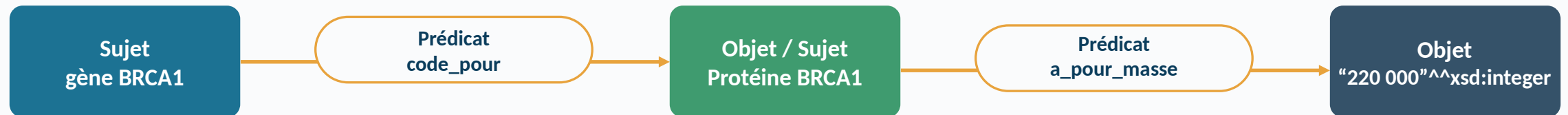
sujet + prédicat + objet $\Leftrightarrow$ head + relation + tail

Définition formelle

$$G = (E, R, L, T)$$

entités **E** · relations **R** · littéraux **L** · triplets $T \subseteq E \times R \times (E \cup L)$

Chaque fait du graphe s'exprime comme un triplet :



Un graphe de connaissances = un ensemble de triplets, typés et reliés par un schéma (ontologie).

sujet + prédicat + objet <=> head + relation + tail

PARTIE 4 • 20 MIN

Modèles de représentation formelle

RDF, RDFS, OWL, property graphs : comment encoder concrètement un graphe de connaissances ?

RDF — Resource Description Framework

- Tout fait est un triplet (sujet, prédicat, objet)
- Chaque ressource est identifiée par un IRI (identifiant global, type URL)
- Les namespaces évitent les ambiguïtés entre vocabulaires
- Plusieurs formats de sérialisation possibles pour un même graphe

Exemple de triplet RDF

`ex:BRCA1_gene`

`ex:codePour`

`ex:BRCA1_protein .`

Formats de sérialisation courants

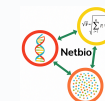
Turtle

RDF/XML

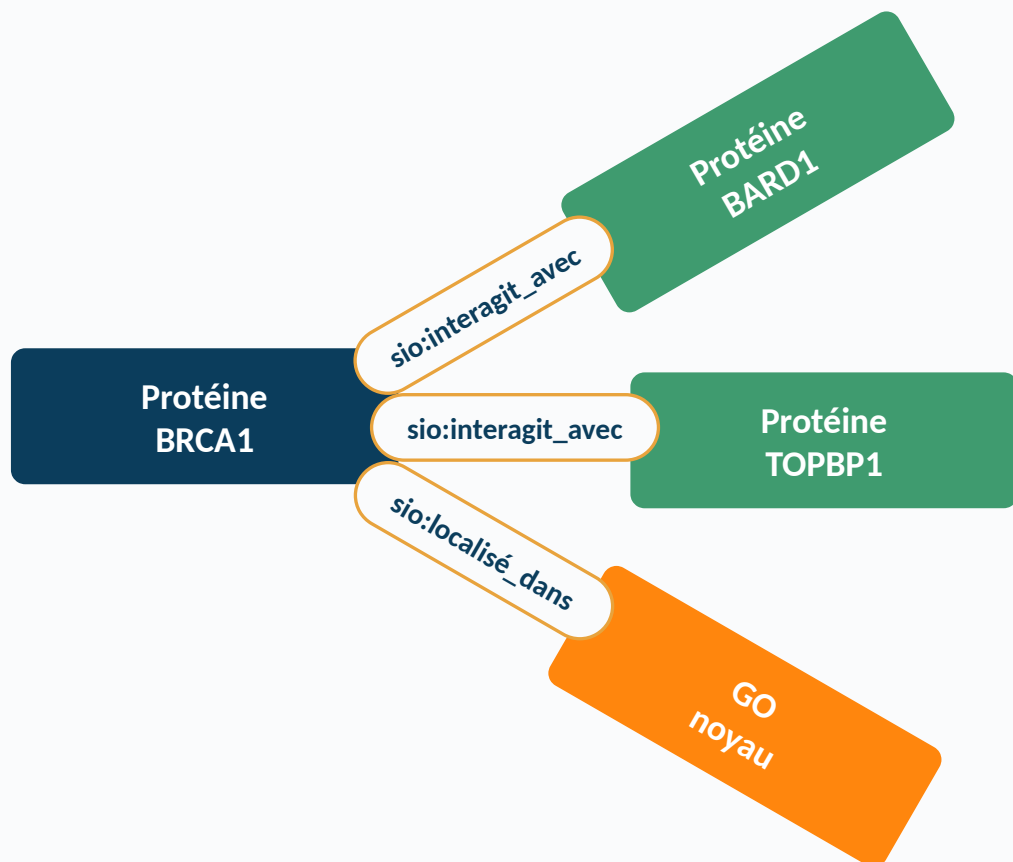
JSON-LD

N-Triples

ex: est un préfixe (namespace) qui abrège un IRI complet.



Exemples de formats de sérialisation



N-Triples

```
@prefix sio: <http://semanticscience.org/resource/SIO_> .
@prefix proteine: <http://proteine.org/#> .
@prefix go: <https://amigo.geneontology.org/amigo/term/GO:> .

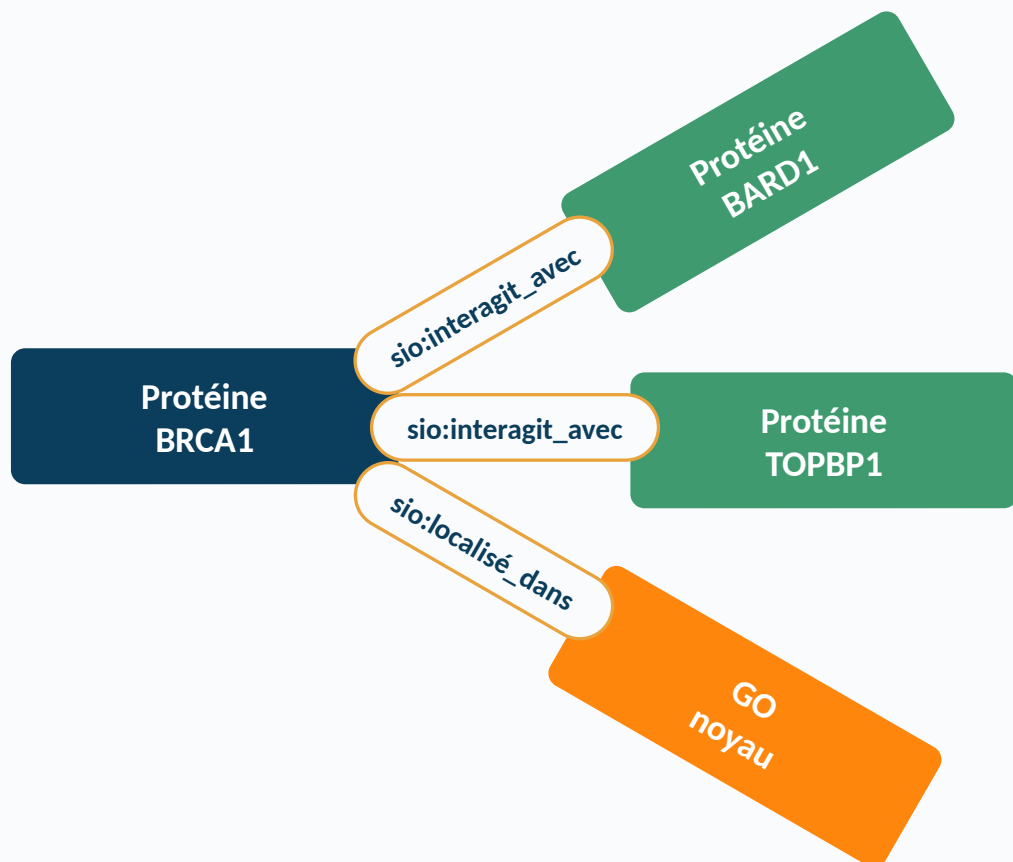
proteine:BRCA1 sio:interagit_avec proteine:BARD1 .
proteine:BRCA1 sio:interagit_avec proteine:TOPBP1 .
proteine:BRCA1 sio:localisé_dans go:noyau .
```

Turtle

```
@prefix sio: <http://semanticscience.org/resource/SIO_> .
@prefix proteine: <http://proteine.org/#> .
@prefix go: <https://amigo.geneontology.org/amigo/term/GO:> .

proteine:BRCA1 sio:interagit_avec proteine:BARD1 ,
                proteine:TOPBP1 ;
                sio:localisé_dans go:noyau .
```

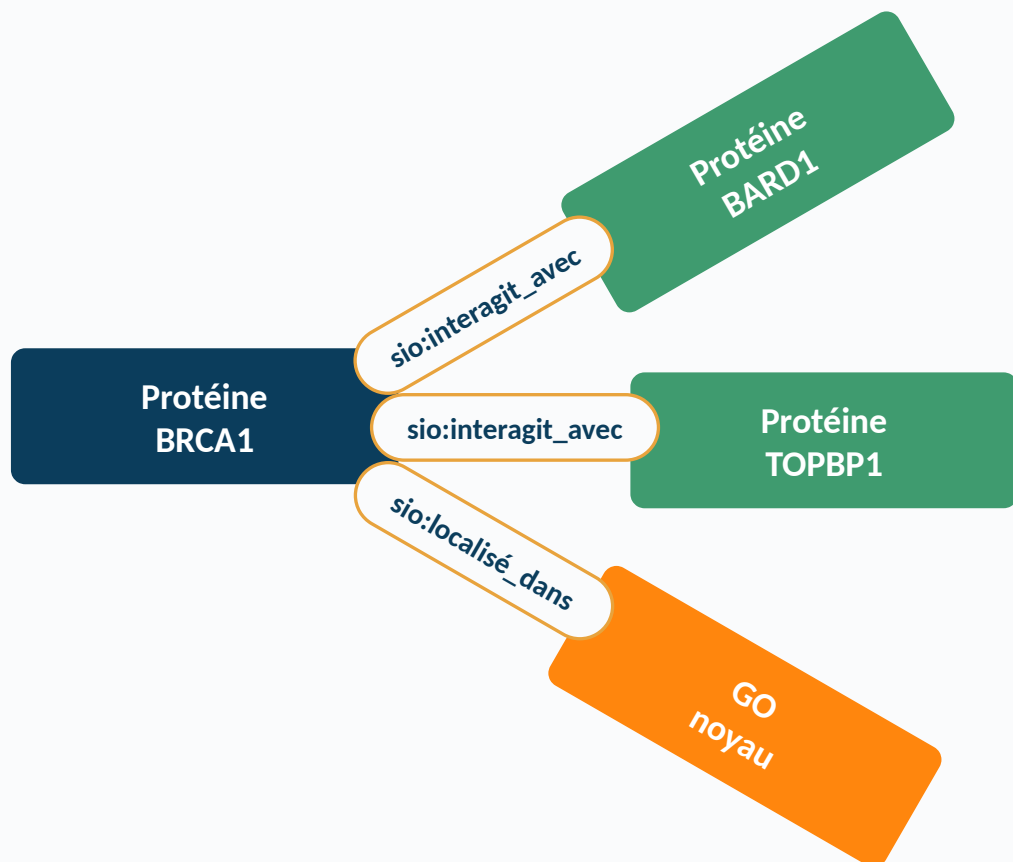
Exemples de formats de sérialisation



RDF/XML

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:sio="http://semanticscience.org/resource/SIO_"
  xmlns:proteine="http://proteine.org/#"
  xmlns:go="https://amigo.geneontology.org/amigo/term/GO:">
  <rdf:Description rdf:about="http://proteine.org/#proteine">
    <sio:interagit_avec rdf:ressource="http://proteine.org/#BARD1"/>
    <sio:interagit_avec rdf:ressource="http://proteine.org/#TOPBP1"/>
    <sio:localisé_dans rdf:ressource="https://amigo.geneontology.org/amigo/term/GO:noyau"/>
  </rdf:Description>
</rdf:RDF>
```

Exemples de formats de sérialisation



JSONLD

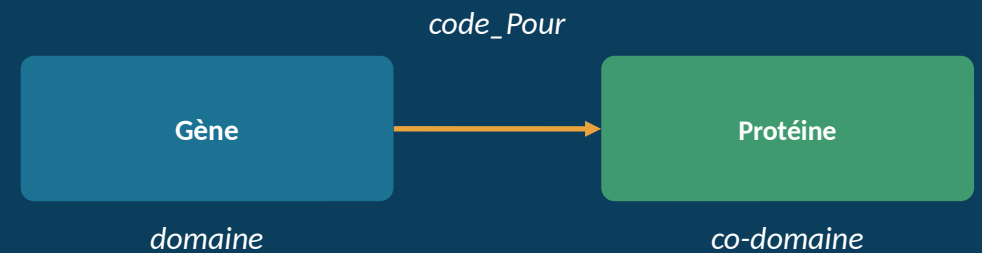
```
{
  "@context": {
    "proteine": { "@id": "http://proteine.org/#", "@prefix": true },
    "sio": { "@id": "http://semanticscience.org/resource/SIO_", "@prefix": true },
    "go": { "@id": "https://amigo.geneontology.org/amigo/term/GO:", "@prefix": true },
    "sio:interagit_avec": { "@type": "@id" },
    "sio:localisé_dans": { "@type": "@id" }
  },
  "@graph": [
    {
      "@id": "proteine:BRCA1",
      "sio:interagit_avec": ["proteine:BARD1", "proteine:TOPBP1"],
      "sio:localisé_dans": "go:noyau"
    }
  ]
}
```

RDFS — RDF Schema

- **Classes** et hiérarchies de classes (`rdfs:subClassOf`)
- **Propriétés** et hiérarchies de propriétés (`rdfs:subPropertyOf`)
- **Domaine** d'une relation (`rdfs:domain`) = qui peut-être sujet d'un prédicat
- **"Range"** (co-domaine) d'une relation (`rdfs:range`) = qui peut être objet d'un prédicat

RDFS reste léger : pas d'axiomes complexes (restrictions, disjonction...) — c'est OWL qui les apporte.

Domaine et co-domaine d'un prédicat



```
rdfs:domain(code_Pour) = Gène  
rdfs:range(code_Pour) = Protéine
```

OWL — classes, restrictions et axiomes

- **Classes** définies par des restrictions (cardinalité, restriction de valeur), pas seulement nommées
- **Équivalence** de classes (deux classes désignent le même ensemble)
- **Disjonction** de classes (deux classes ne peuvent avoir de membre commun)
- Permet de faire **déduire automatiquement** qu'une entité appartient à une classe, sans l'affirmer explicitement

Exemple d'axiome OWL

GèneDeSusceptibilité $\equiv$

Gène $\sqcap$ AssociéÀ.Maladie

« Un gène de susceptibilité EST DÉFINI COMME tout gène associé à au moins une maladie. » Un raisonneur peut alors classer automatiquement tout gène remplissant cette condition — même si le fait n'a jamais été affirmé explicitement.

OWL — propriétés des relations

T

Transitive

Si A part_of B et B part_of C, alors A part_of C

Ex. : noyau part_of cellule part_of tissu

S

Symétrique

Si A interagit_avec B, alors B interagit_avec A

Ex. : interaction protéine-protéine

F

Fonctionnelle

Une entité n'a qu'une seule valeur pour cette relation

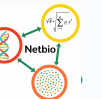
Ex. : a_pour_masse_moléculaire

I

Inverse

Deux relations qui se « lisent » dans les deux sens

Ex. : codePour / estCodéPar



Ontologies, taxonomies, vocabulaires contrôlés



Vocabulaire contrôlé

- Liste fermée de termes normalisés
- Pas de relations formelles entre termes
- Ex. liste de noms de tissus normalisés



Taxonomie

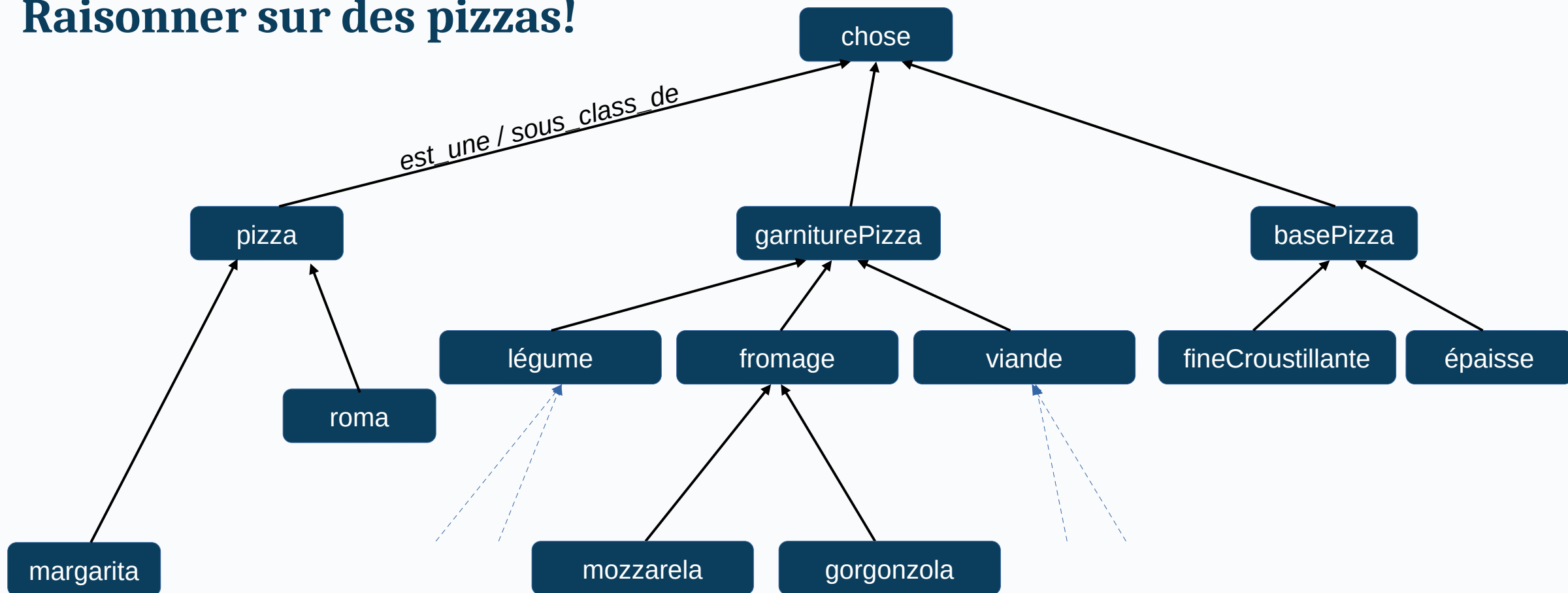
- Hiérarchie simple (is-a)
- Un seul type de relation
- Ex. classification linnéenne des espèces



Ontologie

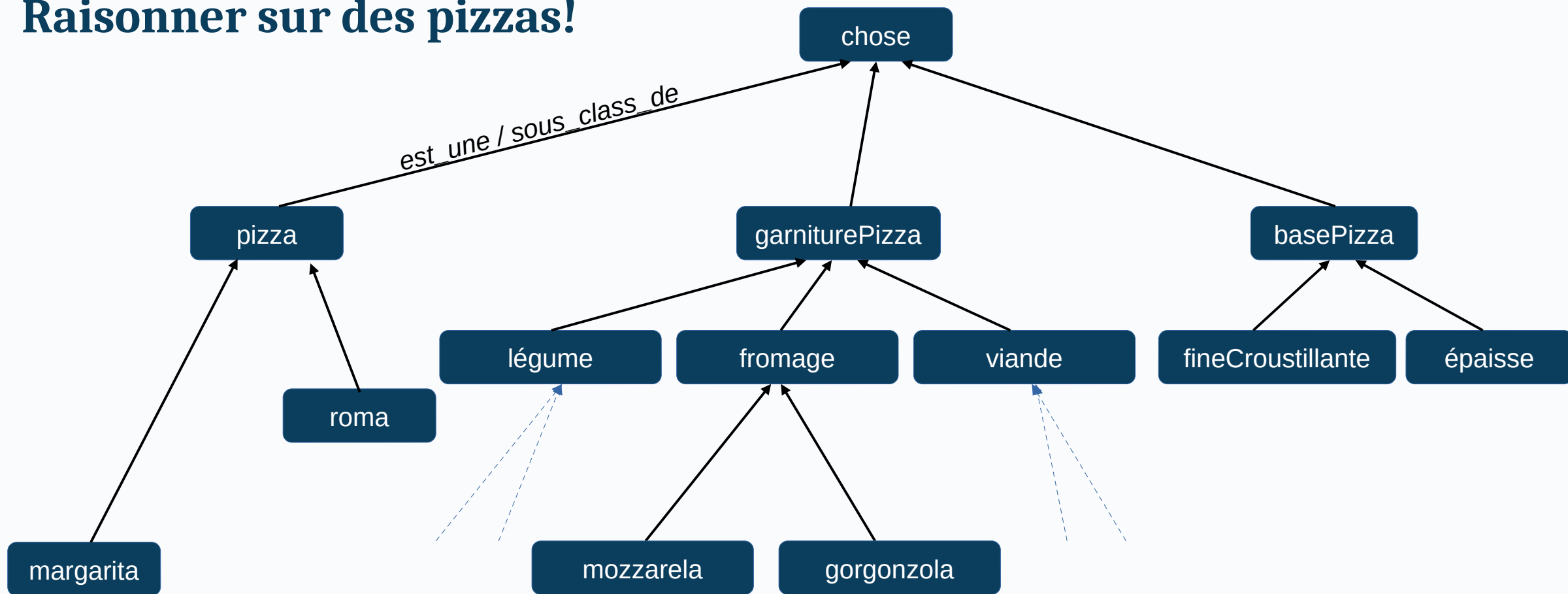
- Relations multiples typées
- Axiomes logiques (OWL)
- Ex. Gene Ontology, ChEBI

Raisonner sur des pizzas!



On définit des classes primitives

Raisonner sur des pizzas!



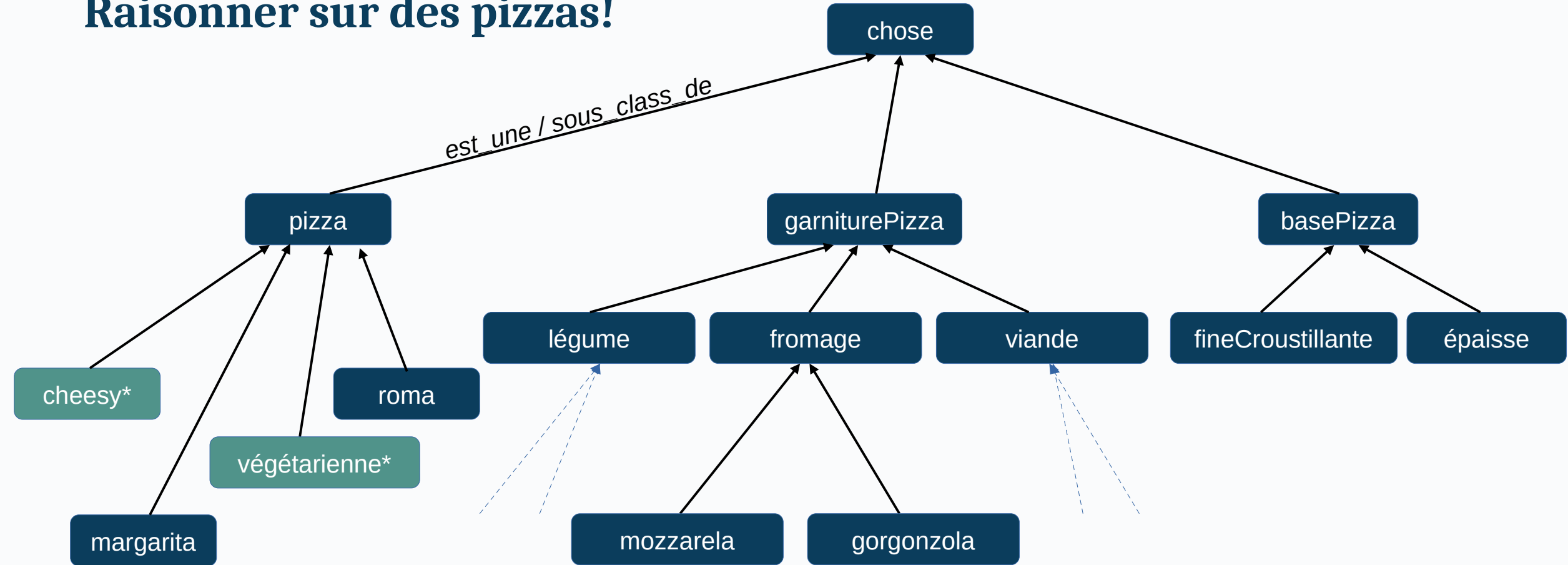
On définit des relations et leurs propriétés

ex: Relation **a_pour_garniture**

Propriétés: domain = pizza, range = garniturePizza

pizza **a_pour_garniture** garniturePizza

Raisonner sur des pizzas!

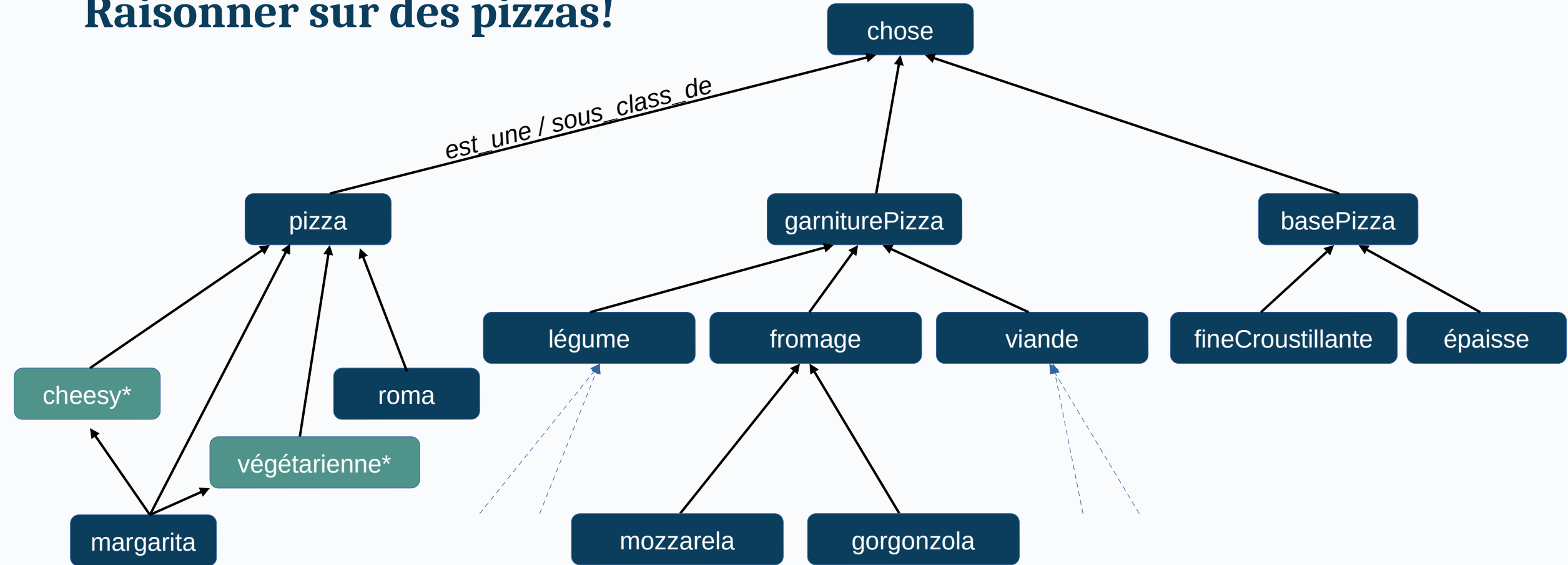


cheesy*, végétarienne* = classes définies = doivent respecter des propriétés

cheesy si (**est_une** pizza et **a_pour_garniture** fromage)

végétarienne si (**est_une** pizza et **non a_pour_garniture** viande)

Raisonner sur des pizzas!



Donc margarita est une pizza cheesy et végétarienne

Implémentation : la Gene Ontology

- Un DAG réparti en 3 namespaces indépendants
- Relations formelles (Relations Ontology) : is_a, part_of / has_part, regulates (+ positively_/negatively_)
- Format natif OBO (texte lisible), mappé vers OWL pour l'interopérabilité
- Annotations via fichiers GAF : gène ↔ terme GO ↔ code d'évidence (21 possibles) ↔ référence
- Membre fondateur de l'OBO Foundry, garantissant l'interopérabilité avec les autres ontologies biomédicales



Biological Process

ex. réparation de l'ADN



Molecular Function

ex. activité hélicase



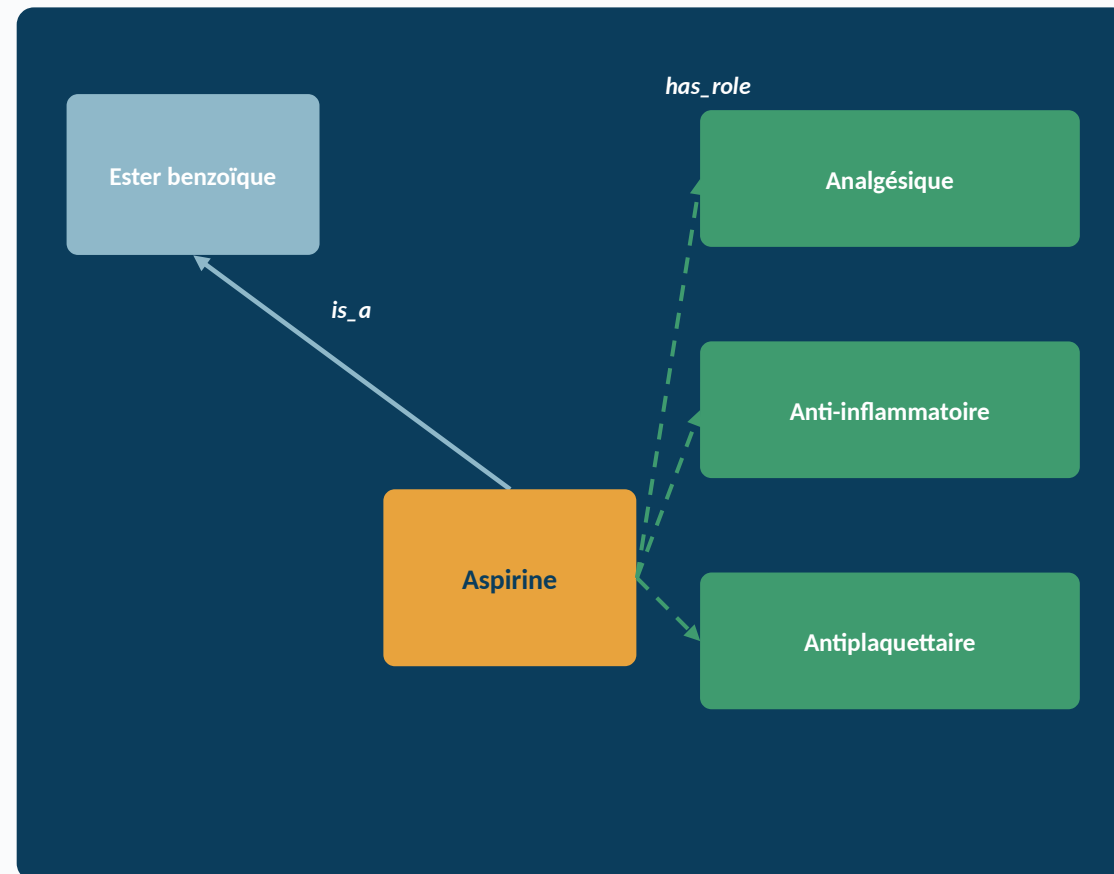
Cellular Component

ex. noyau

Implémentation : ChEBI et la relation has_role

- 3 sous-ontologies : structure moléculaire, rôle, particule subatomique
- Format natif : OWL directement (contrairement à GO, nativement OBO)
- Alignée à BFO (upper ontology) ; plus de 180 000 classes (EMBL-EBI)

L'exemple : is_a / has_role. Une taxonomie (arbre à parent unique) ne peut pas représenter qu'une même molécule joue plusieurs rôles à la fois selon le contexte — c'est ce qu'apporte une vraie ontologie.



https://biportal.bioontology.org/



Welcome to BioPortal, the world's most comprehensive repository of biomedical ontologies

Search for a class

Advanced search

Find an ontology

Browse ontologies

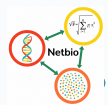
Ontology visits (August 2026)

Ontology	Visits (approx.)
SNOMEDCT	1,560,000
MESH	1,140,000
RXNORM	600,000
DDSS	240,000
NDFRT	240,000

More

Statistics

Ontologies	1,288
Classes	12,255,287
Properties	167,923
Individuals	2,140,859
Mappings	63,036,280



The Netbio logo is centered, featuring the word "Netbio" in a bold, sans-serif font. To its left is a red circular icon containing a white DNA double helix. To its right is a yellow circular icon containing a mathematical formula:
$$\sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}$$
. Below the "Netbio" text is a circular icon with a red border and a light blue background, filled with numerous small, multi-colored dots representing a cell or molecular structure. Green double-headed arrows connect the "Netbio" text to each of these three icons.

https://agrovoc.fao.org/


**Food and Agriculture Organization
of the United Nations**

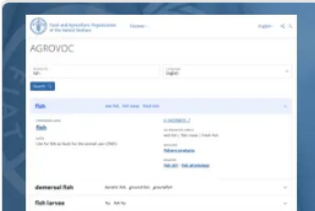
Discover ▾

English ▾




AGROVOC

AGROVOC data can be accessed in different ways depending on the user's needs. If a user wants to browse AGROVOC, search for concepts or terms, or look at the structure, the web browsing interface is the best option. If the user wants to use content of AGROVOC, extract parts of it, or embed it in applications, then the best choice is to download it or to use one of the programmable interfaces, like the SPARQL endpoint or the REST API.



Advanced Search

If a user wants to browse AGROVOC, search for concepts or terms, or look at the structure, the web browsing interface is the best option

[Search Agrovoc >](#)



SKOSMOS

For simple web/based browsing, AGROVOC can use the Skosmos Search & browse interface. It is an open source web-based SKOS browser and publishing tool. This interface offers search and browse functionalities, alphabetical and thematic index, structured concept display, visualized concept hierarchy and multilingual user interface.

[Search SKOSMOS >](#)



SPARQL endpoint

There are other web-based interactive ways to use AGROVOC, like the web-based SPARQL interface which requires specific technical knowledge and the results are in machine-readable formats.

[Query SPARQL >](#)



REST Webservices

Skosmos provides a set of REST APIs to access its vocabulary data.

[Get instructions >](#)

https://cropontology.org/



[About](#) [Guidelines](#) [Site Help](#) [Governance](#) [API](#) [CoP Tools](#) [Contact](#)

[Login](#)

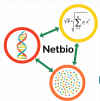
New: All crop-specific ontologies have now a DOI in the open repository Zenodo. It was added in the banner of each ontology page

You can register in the [CoP Social Network](#) to participate to discussions!

Instructions for developing a new ontology or submit new traits to an existing crop are in the 'Guidelines' menu.

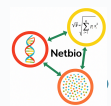
Phenotype and Trait Ontology

Ontology name	Ontology description	Type
Apple Created on Tuesday 23rd of August, 2022. 13:13:10 53 variables	Apple (Malus domestica) defines crop traits and variables to support the standardisation of apple breeding databases providing description of agronomic, morphological, physiological, quality traits, its methods and scales.	Trait
Bambara groundnut Created on Friday 20th of December, 2019. 08:42:44 163 variables	version Dec 2019	Trait
Banana Created on Thursday 18th of April, 2019. 10:45:39 390 variables	Banana Trait Dictionary in template 5 - Bioversity & IITA - April 2019	Trait
Barley Created on Tuesday 9th of June, 2020. 07:45:15 148 variables	ICARDA - TDv5 - Sept 2018	Trait



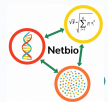
Les données biologiques et les graphes de connaissances

gene1Name	gene2Name	interactionType	detectionMethod	Reference
BRCA1	BARD1	physical	doubleHybrid	pmid:12345



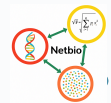
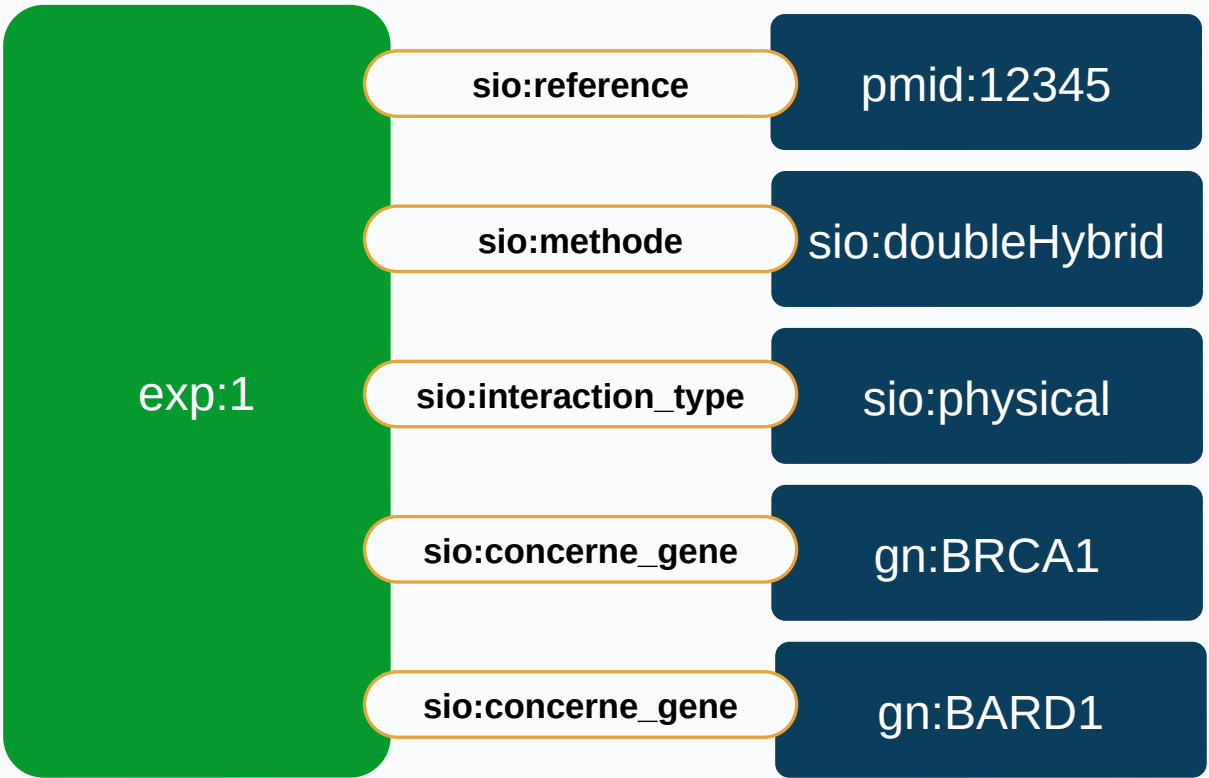
Les données biologiques et les graphes de connaissances

gene1Name	gene2Name	interactionType	detectionMethod	Reference
BRCA1	BARD1	physical	doubleHybrid	pmid:12345



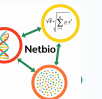
Solution : la réification

gene1Name	gene2Name	interactionType	detectionMethod	Reference
BRCA1	BARD1	physical	doubleHybrid	pmid:12345



Property Graphs

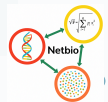
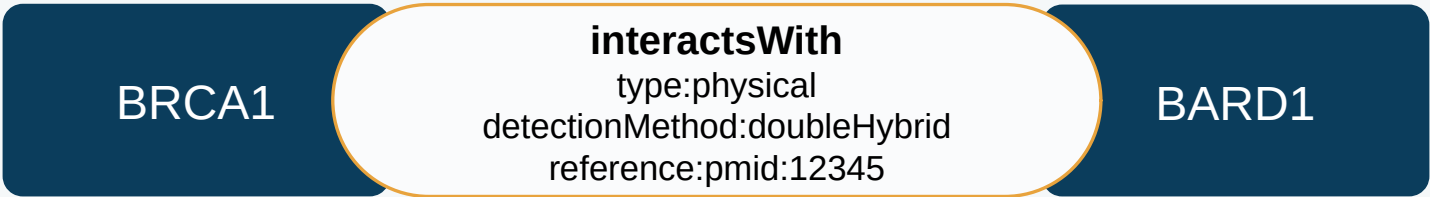
- Noeuds et arêtes portent des attributs libres (clé-valeur)
- Approche « schema-optional » : pas de séparation stricte schéma/données comme en RDF/OWL



Property Graphs

- Noeuds et arêtes portent des attributs libres (clé-valeur)
- Approche « schema-optional » : pas de séparation stricte schéma/données comme en RDF/OWL

gene1Name	gene2Name	interactionType	detectionMethod	Reference
BRCA1	BARD1	physical	doubleHybrid	pmid:12345



Property Graphs

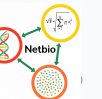
Format	Type	Où on le rencontre
CSV	Tabulaire	Neo4j (import), PrimeKG
GraphML	XML	Gephi, NetworkX
GEXF	XML	Export natif Gephi
GraphSON	JSON	TinkerPop / Gremlin
Cypher	Script	Import/export Neo4j
Cytoscape.js JSON	JSON	Visualisation web
PG	Texte / JSON	Spécification émergente

Property Graphs

PG

Noeud = (ID :Etiquette {Propriétés})

Arêtes = (noeudSource -> noeudCible :typeDeRelation {Propriétés})



Property Graphs

CYPHER

Noeud: `CREATE (ID:Etiquette {Propriétés})`

Arêtes: `CREATE (n1)-[:RELATION {Propriétés}]->(n2)`

Choisir un formalisme



RDF / OWL

- Interopérabilité, standards W3C
- Raisonnement formel
- Idéal pour l'intégration de données publiques



Property Graphs

- Performance d'exploration
- Flexibilité de modélisation
- Idéal pour les applications et l'analyse de graphes

PARTIE 5 • 15 MIN

Construction, interrogation, visualisation, raisonnement

Comment fabrique-t-on un graphe de connaissances, et comment l'interroge-t-on ?

Construire un graphe de connaissances



Un processus itératif : un KG se met à jour et s'enrichit continuellement.

Construire un graphe de connaissances



Un processus itératif : un KG se met à jour et s'enrichit continuellement.

Construire un graphe de connaissances



Un processus itératif : un KG se met à jour et s'enrichit continuellement.

Interroger un graphe : SPARQL et Cypher



Deux langages, une même logique : décrire un motif du graphe que l'on cherche à retrouver.

SPARQL (sur RDF)

```
SELECT ?protein

WHERE {

    ex:BRCA1_gene ex:codePour ?protein .

}
```

« Quelle protéine est codée par le gène BRCA1 ? »

Cypher (sur Property Graph)

```
MATCH (g:Gene {name:'BRCA1'})

    -[:CODE_POUR]->(p:Protein)

RETURN p
```

Même question, exprimée comme un motif de graphe « dessiné » en texte.

Visualiser un graphe de connaissance



Applications seules

Cytoscape, Gephi, Neo4j ...



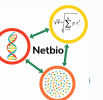
Paquets Python

Netwroxx, igraph, rdflib...



Paquets R

igraph, ggraph, tidygraph...



Visualiser un graphe de connaissance



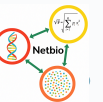
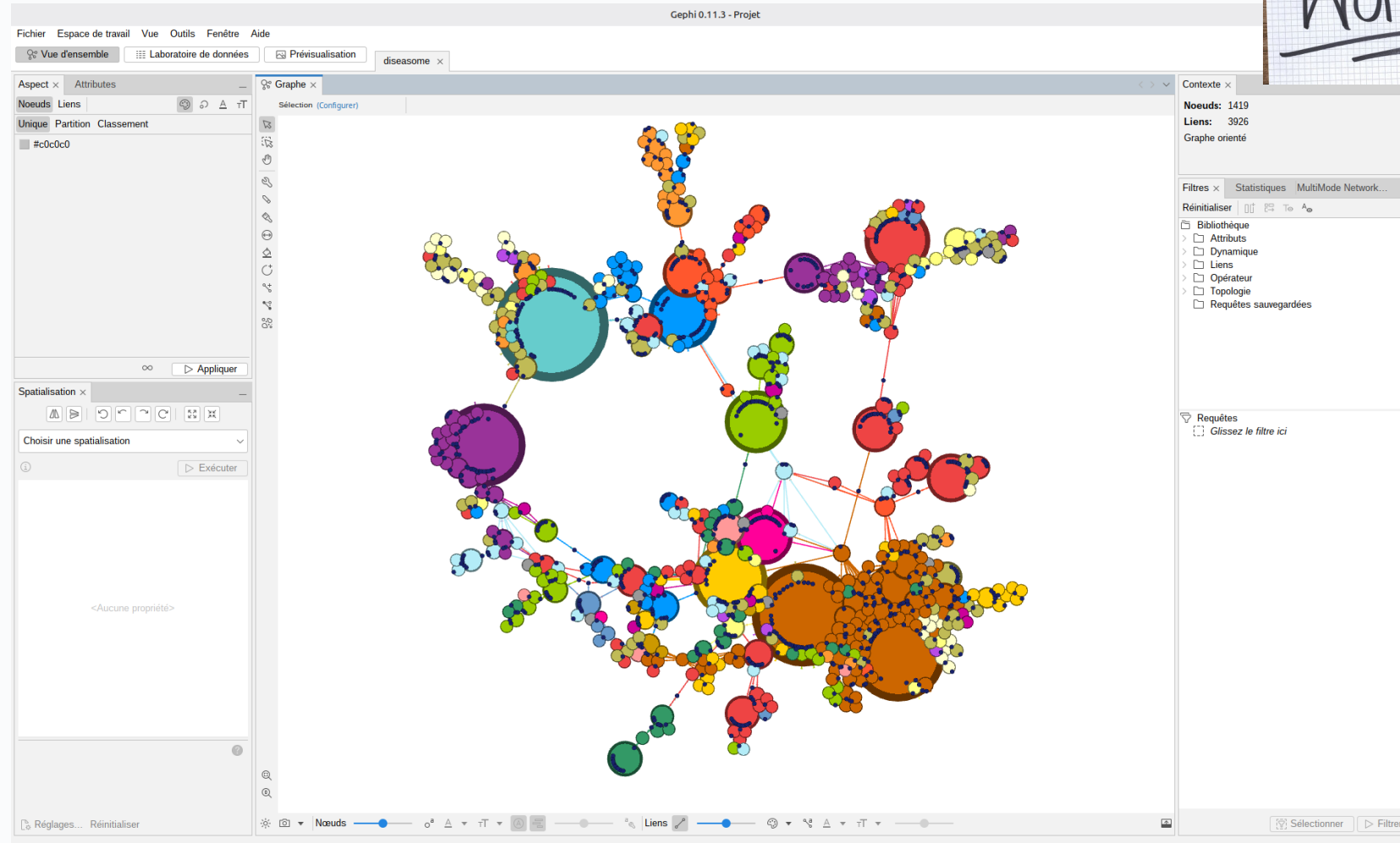
Applications seules

- **Gephi** — référence pour l'exploration de graphes, bons algorithmes de layout et de statistiques réseau intégrés, gère bien les graphes de grande taille.
- **Cytoscape** (desktop) — historiquement pensé pour les réseaux biologiques, très extensible via des apps.
- yEd / yEd Live — éditeur de diagrammes avec layouts automatiques soignés.
- **Neo4j Bloom** et **Neo4j Browser** — exploration visuelle native des graphes de propriétés stockés dans Neo4j, requêtes Cypher avec rendu graphique direct.

Visualiser un graphe de connaissance

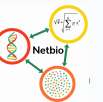
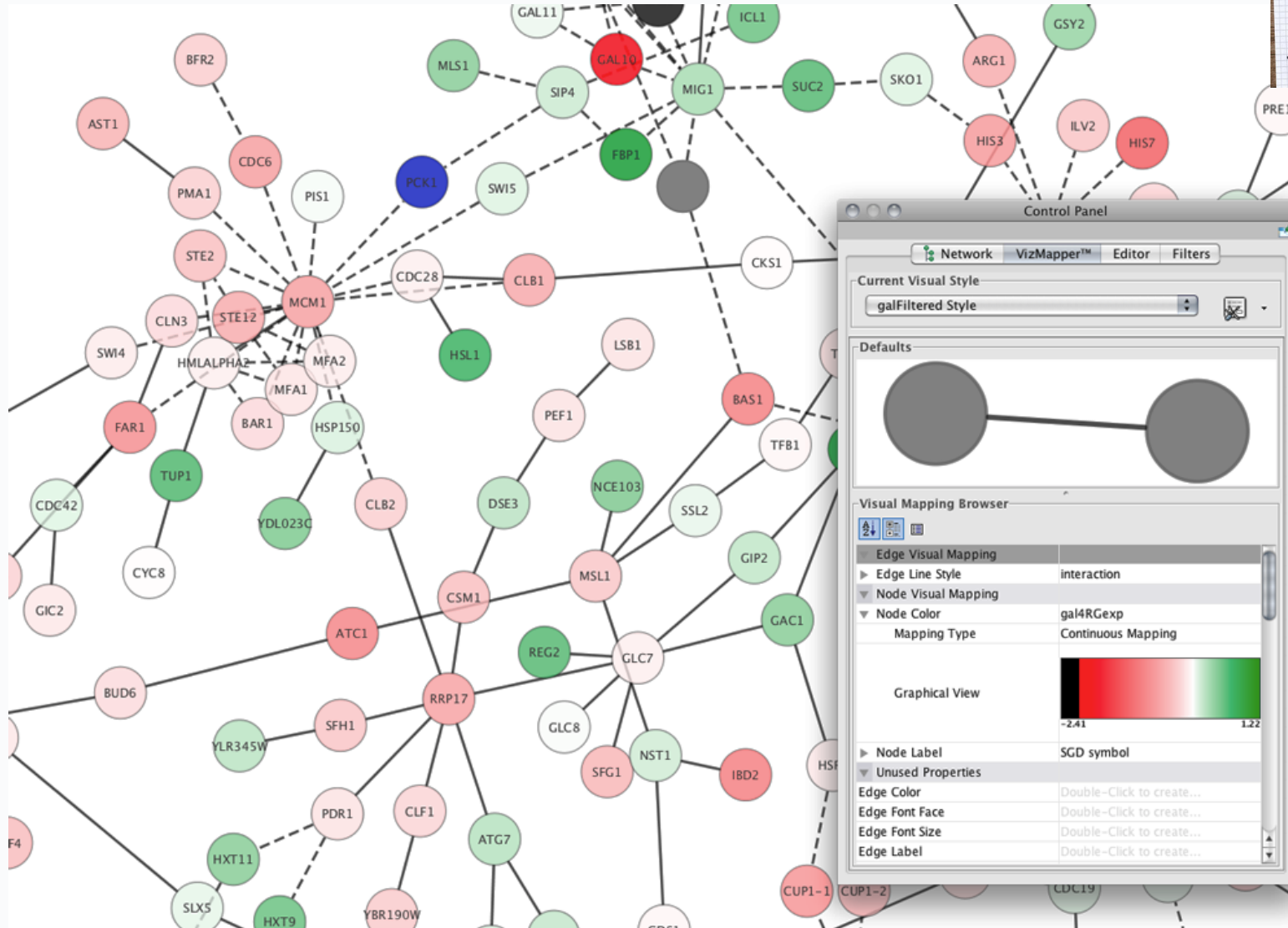


Gephi



Visualiser un graphe de connaissance

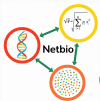
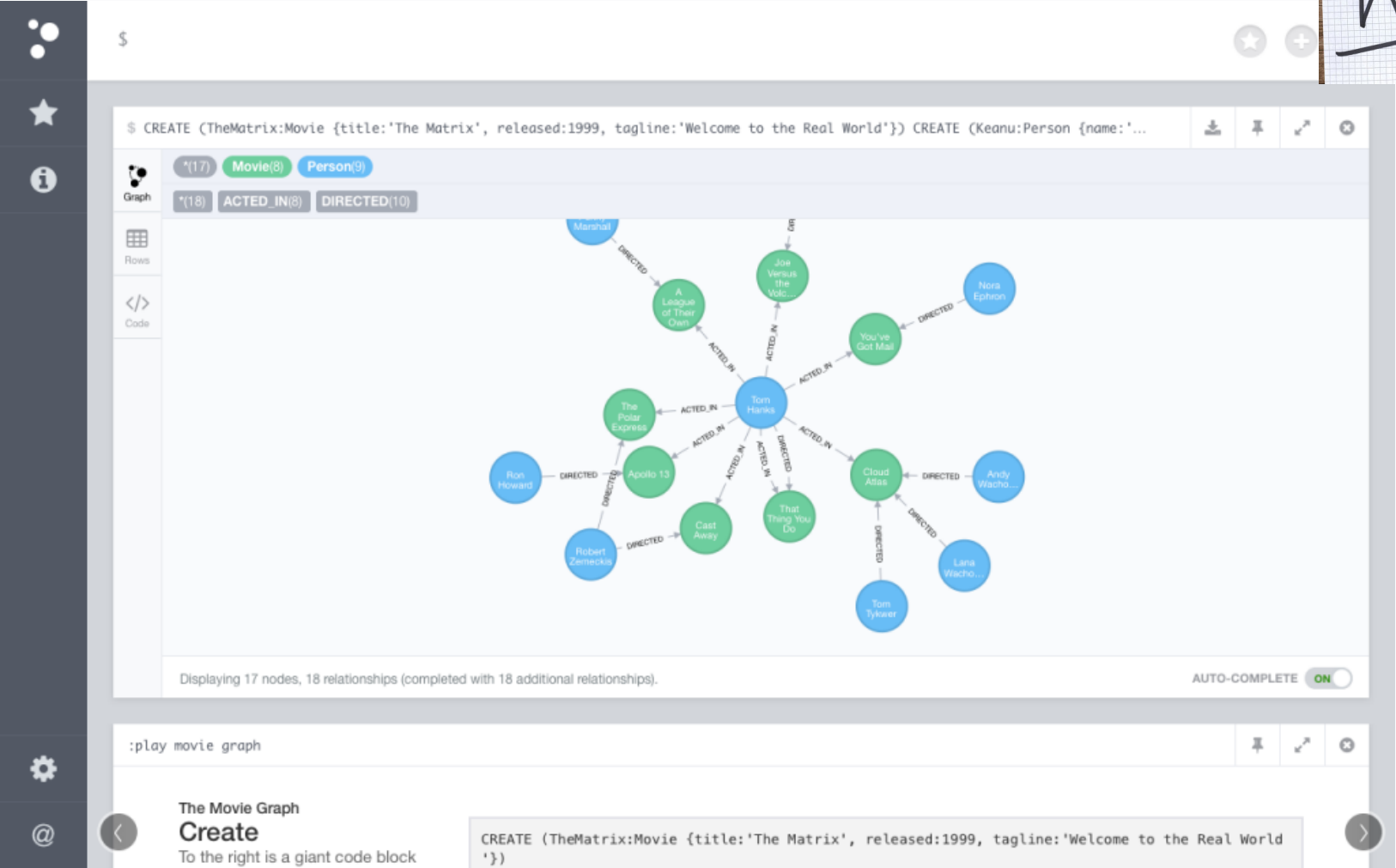
Cytoscape



Visualiser un graphe de connaissance



Neo4j



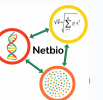
Visualiser un graphe de connaissance



Paquets Python

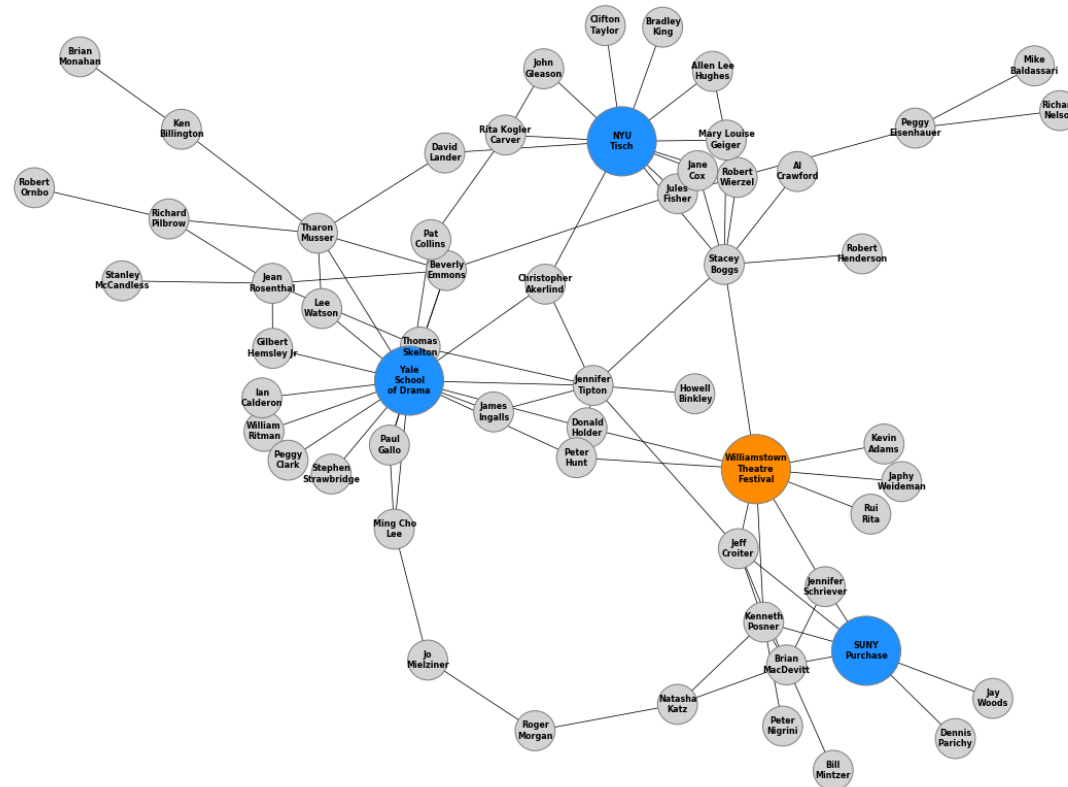
- **networkx** — la base, dessin simple via matplotlib.
- **igraph (python-igraph)** — plus rapide sur de gros graphes, dispose aussi de fonctions de tracé.
- **pyvis** — génère du HTML interactif (basé sur vis.js) à partir d'un graphe networkx, très utilisé en notebook.
- **ipysigma** — widget Jupyter basé sur Sigma.js, gère bien les graphes plus volumineux.

NB: souvent ces paquets nécessite rdflib pour manipuler les données



Visualiser un graphe de connaissance

NetworkX

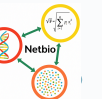


Visualiser un graphe de connaissance



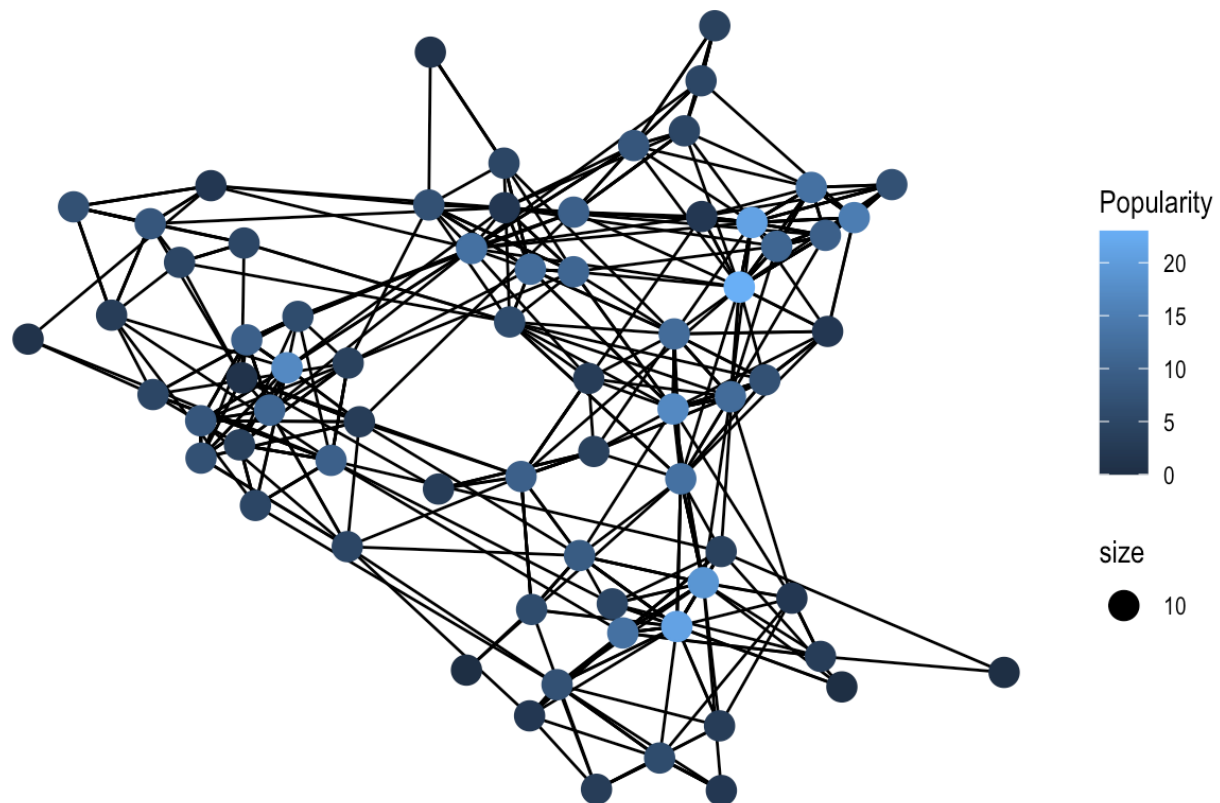
Paquets R

- **igraph** — équivalent R de la librairie Python, cœur de beaucoup de workflows réseau.
- **ggraph** — extension de ggplot2 pour des graphes statiques de qualité publication.
- **tidygraph** — manipulation "tidy" des données de graphe, se combine bien avec ggraph.
- **visNetwork** — htmlwidget interactif basé sur vis.js.



Visualiser un graphe de connaissance

ggraph



Raisonner sur un graphe de connaissances



Inférence

Déduire de nouveaux triplets à partir des axiomes du schéma (ex. transitivité de `is_a`)



Monde ouvert vs monde fermé

En RDF/OWL : l'absence d'un fait n'implique pas sa fausseté (monde ouvert) — à l'inverse des bases de données classiques



Incomplétude

Un KG réel est toujours partiel : c'est justement ce que l'apprentissage (Partie 5) va exploiter

PARTIE 6 • 10 MIN

Embeddings et apprentissage sur les KG

Passer du symbolique au numérique pour compléter et exploiter un graphe incomplet.

Pourquoi apprendre des représentations ?

- Un graphe de connaissances réel est toujours incomplet
- Le raisonnement logique seul ne permet pas de « deviner » un fait manquant sans axiome explicite
- Idée : projeter entités et relations dans un espace vectoriel continu
- Dans cet espace, la géométrie encode la sémantique du graphe

Symbolique → numérique

(BRCA1_gène, code_pour, BRCA1_protéine)

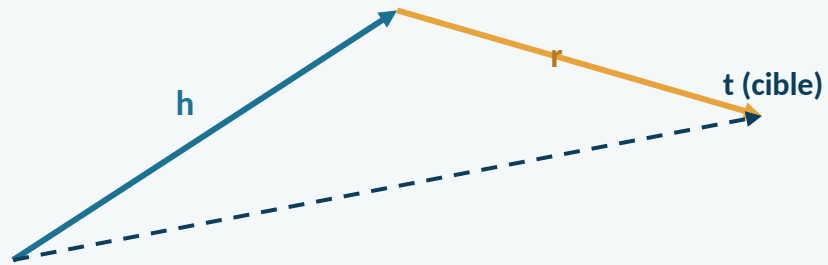
↓ **embedding**

$h(\text{BRCA1_gène}) + r(\text{code_pour}) \approx t(\text{BRCA1_protéine})$

h, r, t sont des vecteurs de faible dimension.

Modèles translationnels et bilinéaires

TransE — intuition géométrique



$$h + r \approx t$$

Score : $d(h + r, t)$ — plus la distance est faible, plus le triplet est plausible.

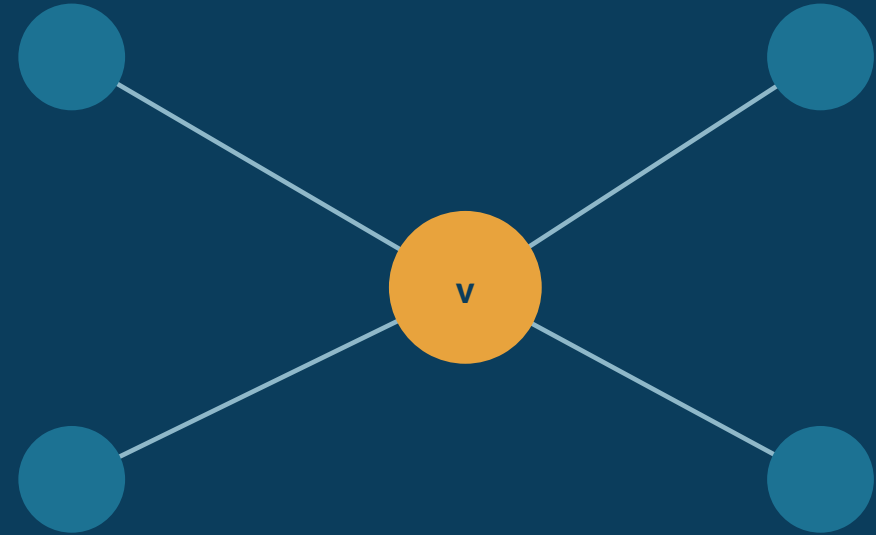
Familles de modèles

- TransE — $h + r \approx t$ (translation)
- DistMult — produit bilinéaire diagonal ; ne capture pas l'asymétrie
- ComplEx — embeddings à valeurs complexes ; capture relations symétriques et antisymétriques

GNN pour graphes hétérogènes : R-GCN, GAT

- Les GNN affinent les embeddings en agrégeant l'information des voisins du graphe
- R-GCN : une transformation différente par type de relation
- GAT : un poids d'attention appris pour chaque voisin (tous les voisins ne comptent pas autant)
- Particulièrement adapté aux KG ayant des relations hétérogènes

Message passing — intuition



Chaque noeud v agrège les messages de ses voisins, pondérés selon le type de relation (R-GCN) ou l'attention apprise (GAT).

Entraîner un modèle d'embedding



Fonction de score

Mesure la plausibilité d'un triplet (h, r, t) — ex. distance ou produit bilinéaire



Negative sampling

On corrompt des triplets vrais (en changeant h ou t) pour créer des exemples négatifs



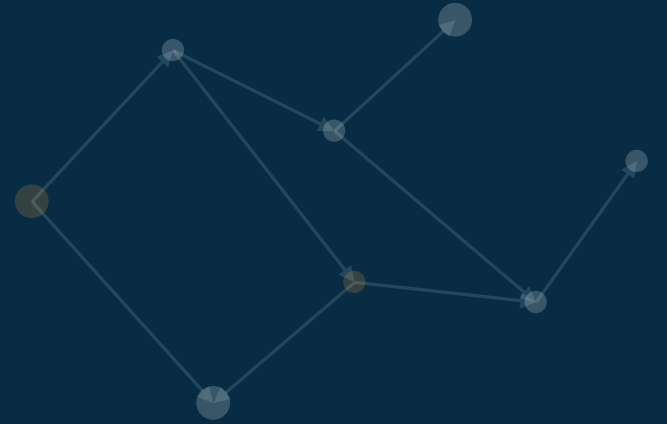
Link prediction

Tâche d'évaluation type : retrouver l'entité manquante dans $(h, r, ?)$ parmi tous les candidats

PARTIE 7 • 10 MIN

Applications en biologie & bio-informatique

Des ressources aux équipes de recherche : un panorama concret du domaine.



Grandes ressources bio-KG



GO

Gene Ontology



UniProt

Protéines



KEGG

Voies métaboliques



Reactome

Réactions & voies



STRING

Interactions protéine-protéine



DisGeNET

Associations gène-maladie



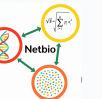
Hetionet

KG intégratif (drug repurposing)



PrimeKG

KG médecine de précision



Graphes de connaissances pour les plantes et l'agronomie



MetaboHUB

- Infrastructure nationale française en métabolomique et fluxomique (INRAE)
- Développe des graphes de connaissances pour contextualiser les données métabolomiques (Metabolomic Semantic Data Lake)
- Graphes FORVM ChemDisease et FORVM Plants : voies toxicologiques, biomarqueurs végétaux



AgroLD

- Agronomic Linked Data (South Green, IRD/CIRAD) — graphe RDF pour les sciences végétales
- Intègre génomique, protéomique et phénotypique sur riz, blé, Arabidopsis...
- Version actuelle : plus d'1 milliard de triplets, 150+ jeux de données

Cas d'usage transversaux



PPI & gène-maladie

Prédiction d'interactions protéine-protéine et d'associations gène-maladie par link prediction



Drug repurposing

Repositionnement de médicaments en combinant KG biomédicaux et embeddings



Intégration multi-omique

Génomique, protéomique, métabolomique reliées dans une même structure interrogeable



Défis spécifiques aux données biologiques

Hétérogénéité

Formats, échelles et vocabulaires très variés selon les sources

Incertitude

Les annotations biologiques sont souvent probabilistes ou contradictoires

Biais

Sur-représentation de certains gènes/organismes très étudiés

Scalabilité

Des graphes à plusieurs millions de noeuds et de relations à traiter

PARTIE 8 • 5 MIN

Synthèse, limites, perspectives

Un bilan rapide, avant de parler du workshop pratique.

Limites actuelles des graphes de connaissances



Qualité & complétude

Les données restent partielles, bruitées, parfois contradictoires selon les sources



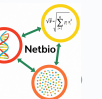
Explicabilité

Les modèles d'embeddings/GNN sont peu interprétables — un enjeu fort en santé



Passage à l'échelle

Des graphes toujours plus grands, à interroger et apprendre efficacement



Tendances récentes : LLM et graphes de connaissances

- RAG sur graphes : récupérer un sous-graphe pertinent pour ancrer la réponse d'un LLM
- Extraction assistée par LLM : construction de KG à partir de texte, plus rapide et plus flexible
- Les deux mondes se complètent : structure et vérifiabilité du graphe + flexibilité du langage naturel

Rendez-vous demain pour le workshop

Ce qui vous attend

- Requêter avec Cypher (Neo4j) et/ou SPARQL (Apache Jena Fuseki)
- Visualiser un graphe de connaissance avec Cytoscape, Gephi ou Neo4j
- Utiliser des méthodes d'embedding sur PrimeKG pour repositionner des drogues (petit dataset)

Merci !

Questions & discussion

Workshop pratique demain →