# Graph Theory

**Dr. Jens Dörpinghaus**

doerpinghaus@uni-koblenz.de

Mathematisches Institut
Universität Koblenz-Landau
Campus Koblenz

Federal Institute for Vocational Education and Training (BIBB)
Robert-Schuman-Platz 3
53175 Bonn

Summer 2022

# 7 Information Networks

- Introduction
- Examples
- Knowledge Graphs
- Analysis and structures of KGs
- Network Dynamics
- Graph Theory and AI
- Technical Remarks

- *Information networks* are networks which involve several sources and/ or several layers.
- Example for such *multilayer networks*:
  - Bibliographic information networks (authors, documents, published venues, or topics).
  - Internet of Things (IoT).
  - Public social networks.
  - Urban transportation networks with spatial locations (places of interest), edges represent vehicles of different types.
- Important: *Data context*.

Tänk! Vem kunde tro att myror

var så fantastiska varelser?

- Data science usually considers some sort basic **data points**.

Format: Text
Source: OCR

Tänk! Vem kunde tro att myror

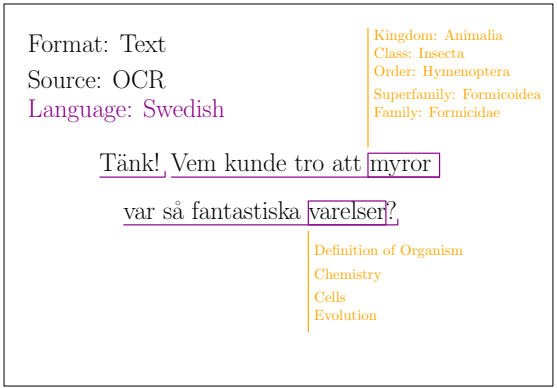
var så fantastiska varelser?

- Data science usually considers some sort basic **data points**.
- **Metadata** helps to understand the *what* and *from where*.

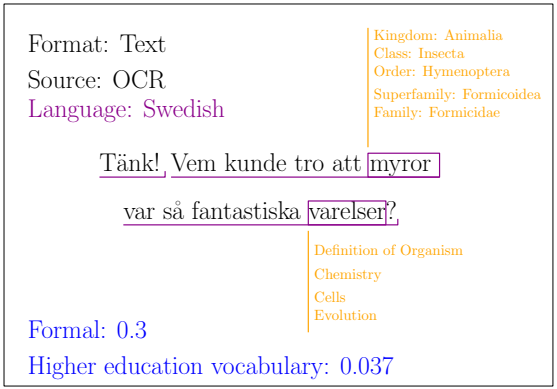
Format: Text
Source: OCR
Language: Swedish

Tänk! Vem kunde tro att myror
var så fantastiska varelser?

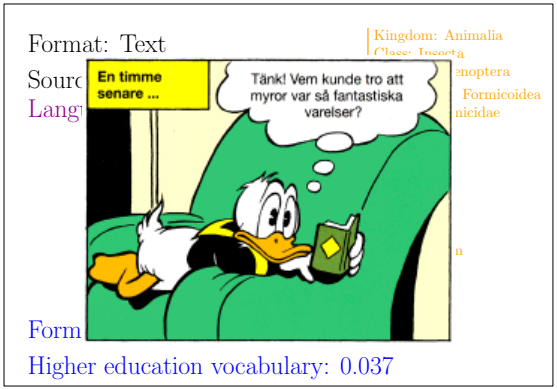
- Data science usually considers some sort basic **data points**.
- **Metadata** helps to understand the *what* and *from where*.
- Basic **data analysis** helps to understand more details of *how* and *what*.




- Data science usually considers some sort basic **data points**.
- **Metadata** helps to understand the *what* and *from where*.
- Basic **data analysis** helps to understand more details of *how* and *what*.
- **Knowledge discovery** and **data mining** helps to understand *the big picture* and broader *how*.




- Data science usually considers some sort basic **data points**.
- **Metadata** helps to understand the *what* and *from where*.
- Basic **data analysis** helps to understand more details of *how* and *what*.
- **Knowledge discovery** and **data mining** helps to understand *the big picture* and broader *how*.
- **Score-based ranks** from TM or other data mining methods help to understand even more...

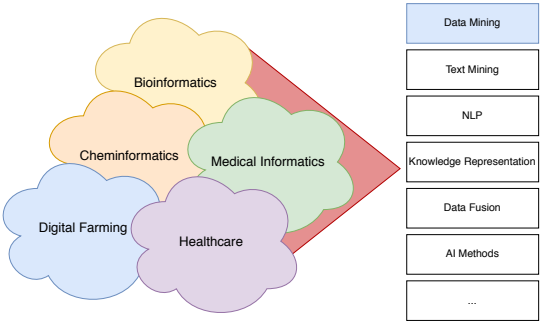



- Data science usually considers some sort basic **data points**.
- **Metadata** helps to understand the *what* and *from where*.
- Basic **data analysis** helps to understand more details of *how* and *what*.
- **Knowledge discovery** and **data mining** helps to understand *the big picture* and broader *how*.
- **Score-based ranks** from TM or other data mining methods help to understand even more...
- To get a good understanding of data and methods, we need to consider every possible context.

# Different Perspectives: Examples from Healthcare


- **Digital transformation** in medicine and healthcare is an emerging topic (e.g. eHealth and personalized medicine).
- **Data science** points at several very challenging topics.
- Reasonable **Data Prediction** is only possible while having all data available.
- Making data **interoperable** and **accessible** is necessary to develop next-generation services.

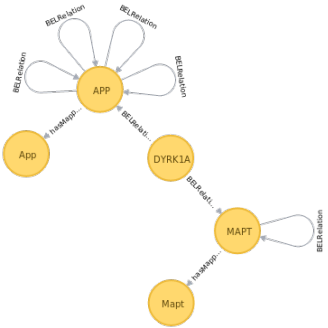
# Different Perspectives: Life Sciences




Even **more data** and **more challenges**:

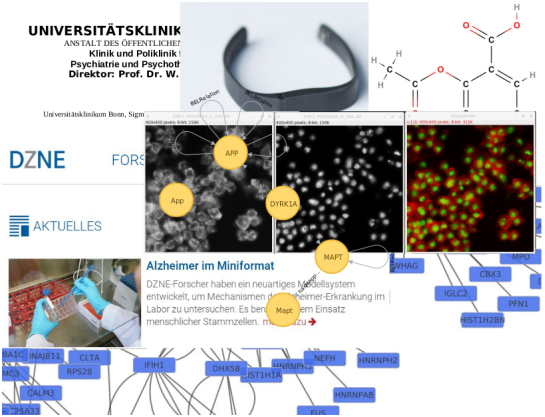
- **Bioinformatics** (e.g. Protein-interaction networks, pathways), **Cheminformatics** (e.g. Molecular geometry), **Digital farming** (e.g. Industry 4.0), ...
- Cross-section tasks:
  - **Text Mining**
  - **Knowledge representation**.
  - **Data fusion**.
  - **AI** methods for analysis and prediction.
  - ...

# Problem Summary



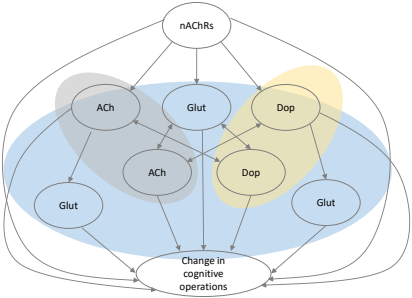

- Data management and analysis is usually **dedicated to a few purposes**.
- Data management **without interoperable data**.
- Challenge: Different technologies, content and standards.
- Generating high-throughput workflows to connect real-world data for predictive purposes **needs context aware technologies**.

1. Data is often not **f**indable.
2. Data is often not **a**ccesible.
3. Data is often not **i**nteroperable.
4. Data is often not **r**e-usable.




Research objectives for current information networks:

- **Knowledge representation**: Methods to build interoperable data and annotate context information.
- Research on **algorithms and concepts** that can be applied on connected data for knowledge discovery.
- Building a state-of-the-art distributed (microservice) **architecture to support digital transformation**.
- Create a proof-of-concept and a generalization which can be used in all data-driven fields.




**Text Mining**

- Context-aware semantic AI (classification, relation extraction, ...)

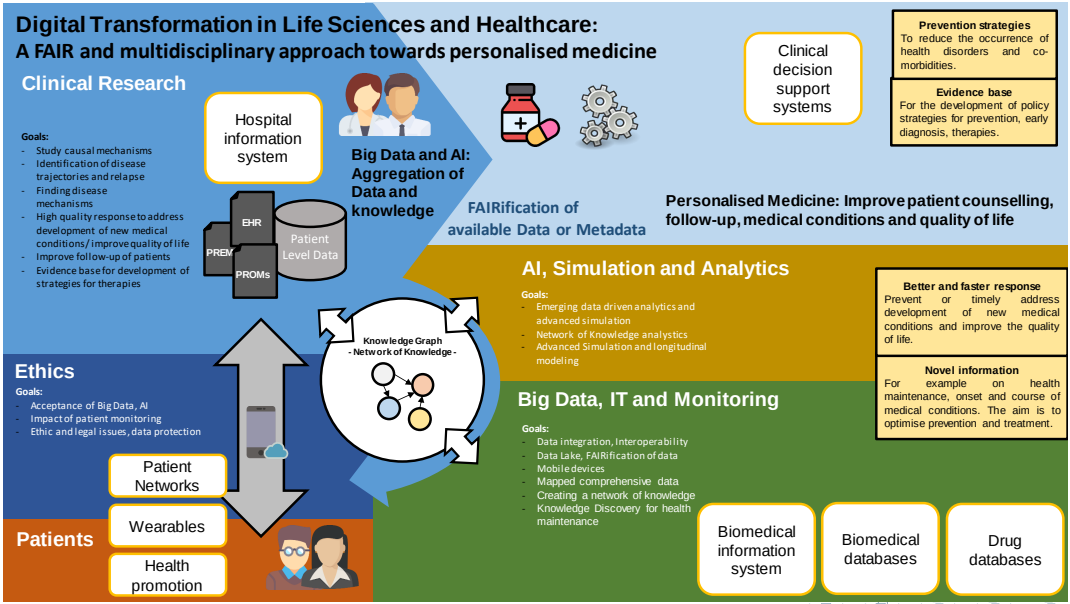
**Simulation**

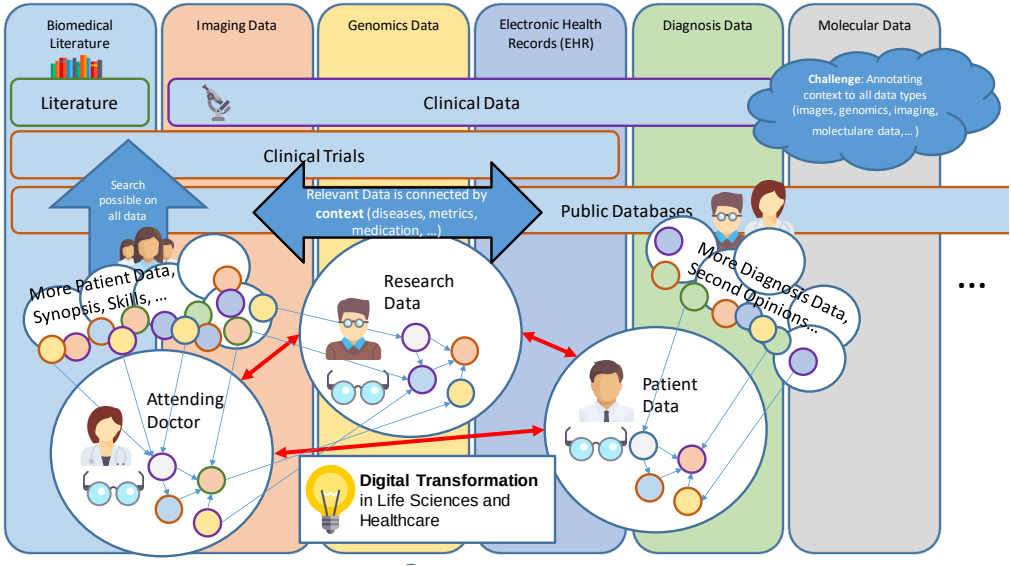
- Context aware systems allow an easy transformation to finite state machines and markovchain models.
- A systematic connection with real world results in easy to use data simulation, real-time predictions and industry 4.0 solutions.

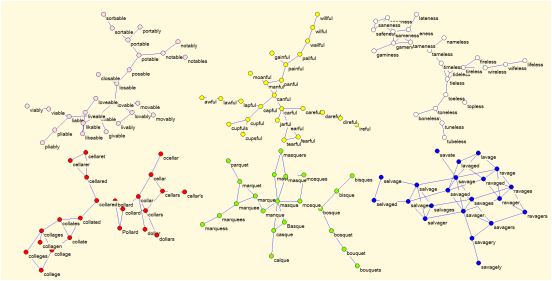
# Applications: Reproducible AI (or: where is the data?)


https://xkcd.com/1838/

- Models are only applicable on **comparable data** (same features, similar value ranges)
- *Example*: Validation of disease risk models → finding independent progression cohort studies
  - It remains unclear if AI models build based on one study generalize.
  - Clinical studies, usually each with own assumptions and biases.
  - External clinical or cohort study data not available.
- Comparing two AD studies: Comparable subcohorts may be found, but the absence of features and especially data is a challenge!
- “Is reproducibility as simple as just running the code?”, see Carter (2019).
- **Perspective: Finding the data** would be a first step towards reproducible AI, making it accesible and interoperable the next one...

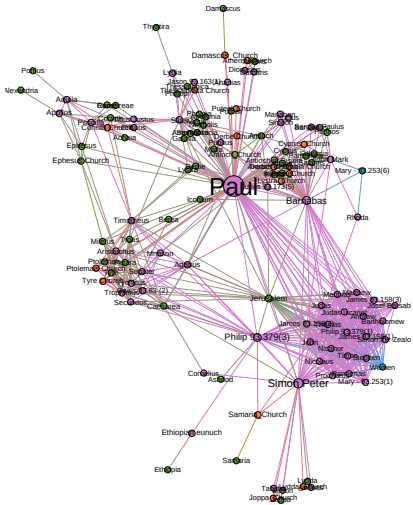

Digital Transformation in Life Sciences and Healthcare:
A FAIR and multidisciplinary approach towards personalised medicine
Clinical Research
Goals:
- Study causal mechanisms
- Identification of disease trajectories and relapse
- Finding disease mechanisms
- High quality response to address development of new medical conditions / improve quality of life
- Improve follow-up of patients
- Evidence base for development of strategies for therapies
Hospital information system
EHR
PREM
PROMs
Patient Level Data
Big Data and AI: Aggregation of Data and knowledge
FAIRification of available Data or Metadata
Clinical decision support systems
Prevention strategies
To reduce the occurrence of health disorders and co-morbidities.
Evidence base
For the development of policy strategies for prevention, early diagnosis, therapies.
Personalised Medicine: Improve patient counselling, follow-up, medical conditions and quality of life
AI, Simulation and Analytics
Goals:
- Emerging data driven analytics and advanced simulation
- Network of Knowledge analystics
- Advanced Simulation and longitudinal modeling
Better and faster response
Prevent or timely address development of new medical conditions and improve the quality of life.
Novel information
For example on health maintenance, onset and course of medical conditions. The aim is to optimise prevention and treatment.
Knowledge Graph
- Network of Knowledge -
Ethics
Goals:
- Acceptance of Big Data, AI
- Impact of patient monitoring
- Ethic and legal issues, data protection
Big Data, IT and Monitoring
Goals:
- Data integration, Interoperability
- Data Lake, FAIRification of data
- Mobile devices
- Mapped comprehensive data
- Creating a network of knowledge
- Knowledge Discovery for health maintenance
Patient Networks
Wearables
Health promotion
Patients
Biomedical information system
Biomedical databases
Drug databases


Biomedical Literature
Imaging Data
Genomics Data
Electronic Health Records (EHR)
Diagnosis Data
Molecular Data
Literature
Clinical Data
Clinical Trials
Challenge: Annotating context to all data types (images, genomics, imaging, molecular data,...)
Search possible on all data
Relevant Data is connected by context (diseases, metrics, medication, ...)
Public Databases
More Patient Data, Synopsis, Skills, ...
Research Data
More Diagnosis Data, Second Opinions...
...
Attending Doctor
Patient Data
Digital Transformation in Life Sciences and Healthcare

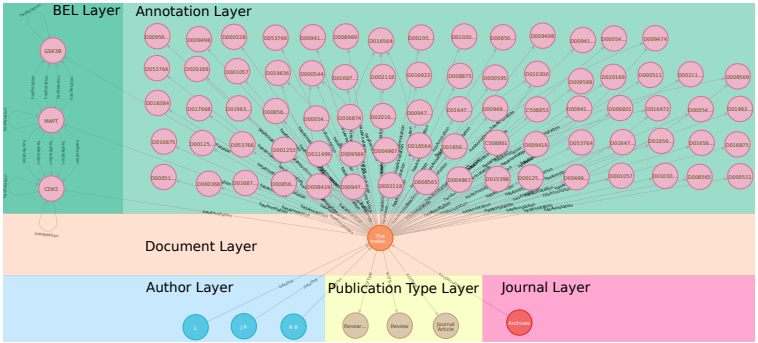


Source: http://mrvar.fdv.uni-lj.si/pajek/XXLexample1.htm

- **Search and Knowledge Discovery** technologies may generalize for all domains.
- **Text Mining and NLP** may be useful especially for text- or literature based domains (e.g. CLARIN, Europeana).
- Combining data with **geographic information** (so called *spatial humanities*), for example in archaeology.
- **Perspective:** Making data from different domains interoperable.




- **Social Network Analysis** (or: *Historical Network Analysis*) can be extended with data networks.
- Example: Combining literature-based networks with external texts and historical information in theology.
- **Perspective**: Combining lessons and models build upon current networks and data and making them interoperable with historical or literature-based networks.


BEL Layer
Annotation Layer
GSK3B
MAPT
CDK5
Document Layer
Author Layer
Publication Type Layer
Journal Layer
Review
Archives

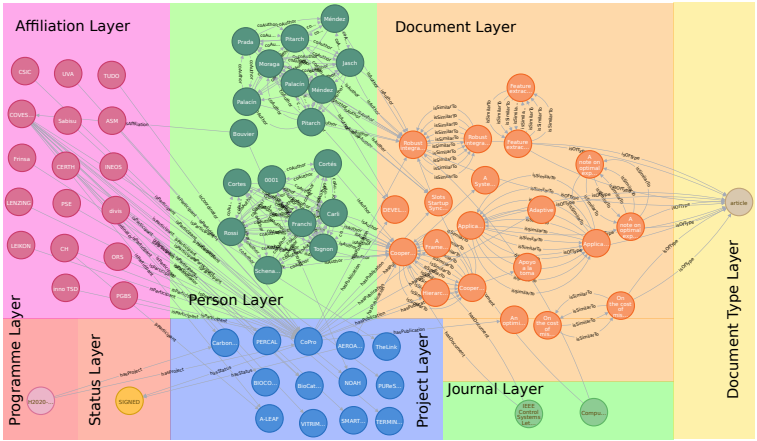



Figure 14: (Illustration of some knowledge graph layers found in the testing environment. Here, we can see some document-specific layers which are combined from several data sources (PubMed, DBLP, H2020): Document Type Layer, Journal Layer, Person (Author) Layer. Other layers are specific to the H2020 project data obtained from EU Open Data Portal: Project Layer, Status Layer, Programme Layer and the Affiliation Layer. We notice several intersections, for example `Quentin_Bouvier` is no Author, but has both an affiliation and is associated with the project `NOAH`.

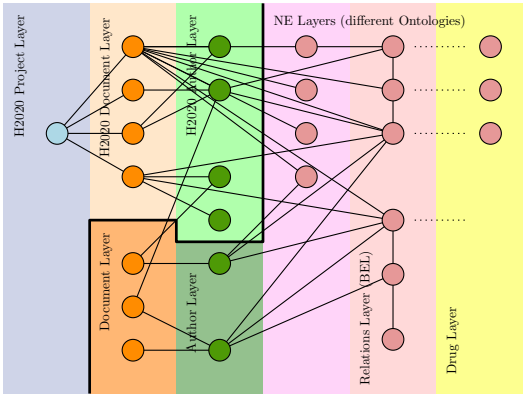
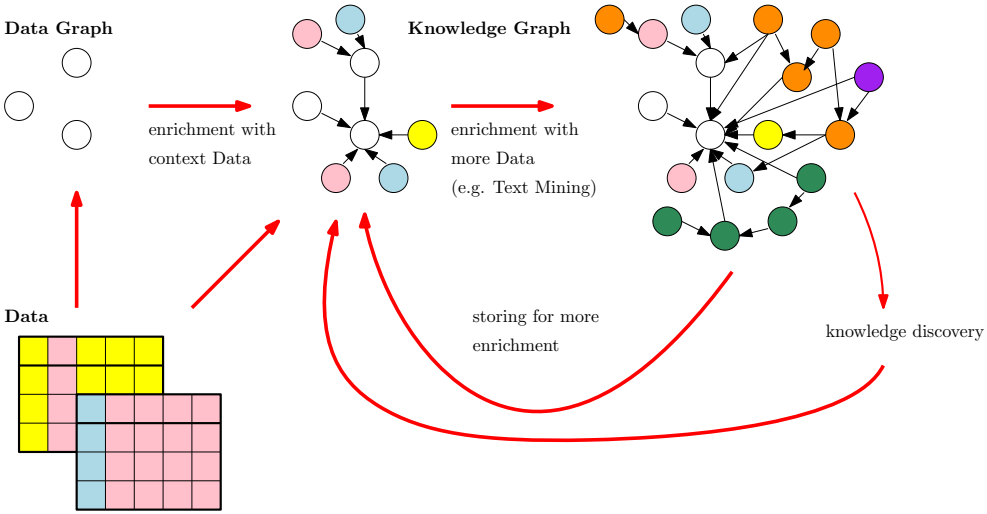


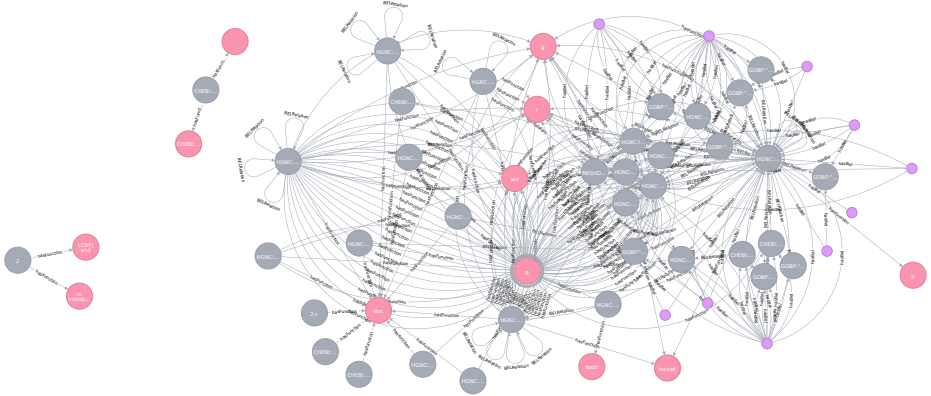

Figure 15: An illustration of the different layers involved in exploring H2020 data. The first layer – H2020 projects – is just contained in H2020 data. Documents and Authors both contain data from H2020 and other sources. All other layers contain data from different ontologies and terminologies. They are connected using NLP and text mining technologies and also contain intra-ontologie relations like biological or cause-and-effect relations.

# Method – Overview



## Definition 7.1 (Knowledge Graph)

We define a knowledge graph as graph $G = (E, R)$ with entities $e \in E = E_1 \cup E_2 \cup ... \cup E_n$ coming from formal structures $E_i$ like ontologies and relations R. We define $R = \{R_1, ..., R_n\}$ as list of either inter-ontology and intra-ontology relations. Both $E$ as well as $R$ are finite discrete spaces.



- Every entity $e \in E$ may have some additional metainformation which needs to be defined with respect to the application of the knowledge graph.
- For instance, there may be several node sets (some ontologies, some actors (employees, stakeholders, ...), locations, ...) $E_1, ..., E_n$ so that $E_i \subset E$ and $E = \cup_{i=1,...,n} E_i$.
- The same holds for $R$ when several context relations come together such as ”is relative of”, ”has business affiliation”, ”has visited”, etc.

## Definition 7.2 (LPG, Lekschas 2015)

A labeled property graph $G = (V, E)$ consists of a set of nodes $V$ (sometimes called knowledge subjects) and edges $E$ (sometimes called links). An edge is always related to exactly two nodes with a fixed direction from a start to an end node, defining the property graph as a directed graph. Both, nodes and edges, can store a set of key-value pairs, called properties and nodes can be tagged with labels additionally.

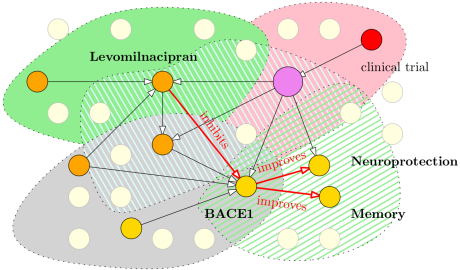
- By using formal structures within the graph, we are implicitly using the model of a labeled property graph.
- Nodes and edges can be identified by using a single or multiple labels, for example using a mapping

$$\lambda : E \cup V \to \Sigma.$$

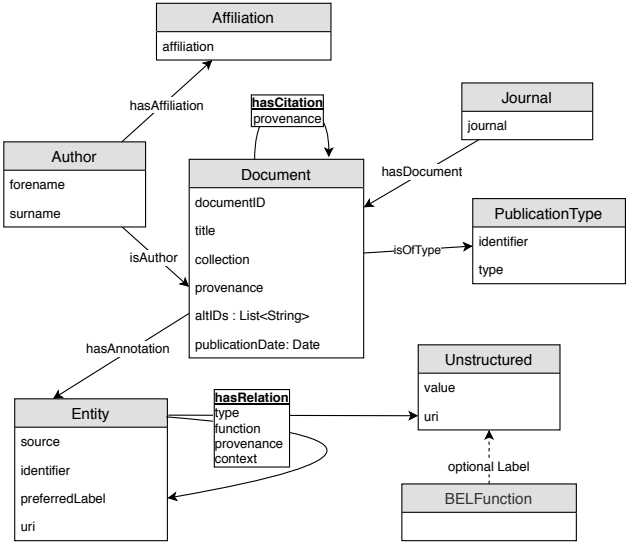
## Definition 7.3 (Context)

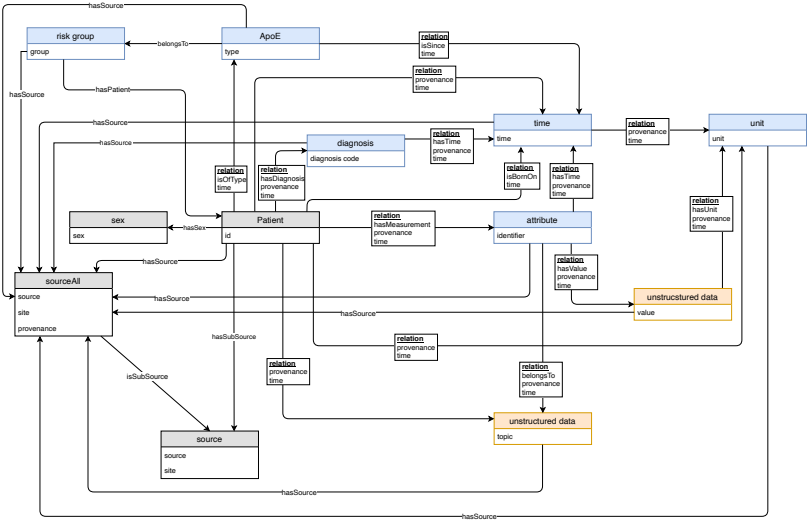
We define a finite, discrete set $C = \{c_1, ..., c_m\}$ of contexts $C_i$. Every node $e \in G$ and every edge $r \in R$ may have one ore more contexts $c \in C$ denoted by $con(e)$ or $con(r)$. It is also possible to set $con(e) = \emptyset$. Thus, we have a mapping $con : E \cup R \to \mathcal{P}(C)$ to the power set of $C$.

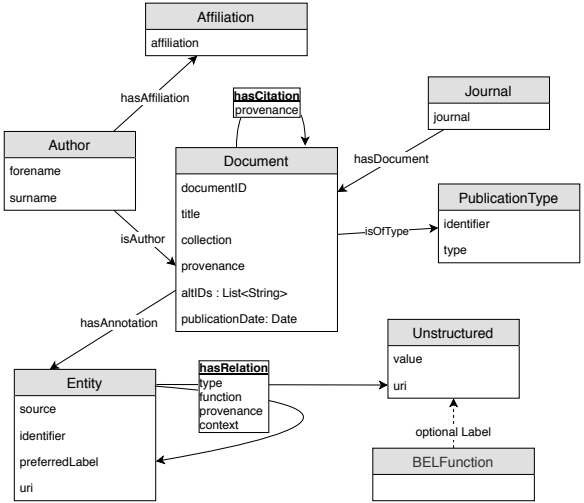
If we use a quite general approach towards context, **we may set $C = E$ and $con(G') = N(G')$. Thus, every inter-ontology *and* intra-ontology relation defines context of two entities.**

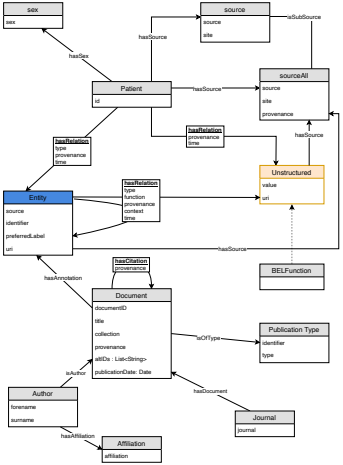


- To build a Knowledge Graph, we need a *Data Schema*.
- It is often convenient to draw this data schema instead of providing a formal definition.
- Different data schemata can be merged to provide a global data schema.




hasSource
risk group
group
belongsTo
ApoE
type
relation
isSince
time
relation
provenance
time
hasSource
hasPatient
hasSource
time
relation
provenance
time
unit
unit
hasSource
diagnosis
diagnosis code
relation
hasTime
provenance
time
relation
isOfType
time
relation
hasDiagnosis
provenance
time
relation
isBornOn
time
relation
hasTime
provenance
time
relation
hasUnit
provenance
time
sex
sex
hasSex
Patient
id
relation
hasMeasurement
provenance
time
attribute
identifier
relation
hasValue
provenance
time
hasSource
sourceAll
source
site
provenance
hasSource
unstrucstured data
value
hasSource
hasSubSource
relation
provenance
time
relation
provenance
time
isSubSource
relation
belongsTo
provenance
time
unstructured data
topic
source
source
site
hasSource
hasSource


Affiliation
affiliation
hasAffiliation
hasCitation
provenance
Journal
journal
Author
forename
surname
Document
documentID
title
collection
provenance
altIDs : List<String>
publicationDate: Date
hasDocument
PublicationType
identifier
type
isAuthor
isOfType
hasAnnotation
Entity
source
identifier
preferredLabel
uri
hasRelation
type
function
provenance
context
Unstructured
value
uri
optional Label
BELFunction


sex
sex
hasSex
source
source
site
hasSource
isSubSource
Patient
id
hasSource
sourceAll
source
site
provenance
hasRelation
type
provenance
time
hasRelation
provenance
time
hasSource
Unstructured
value
uri
Entity
source
identifier
preferredLabel
uri
hasRelation
type
function
provenance
context
time
hasSource
BELFunction
hasAnnotation
hasCitation
provenance
Document
documentID
title
collection
provenance
altIDs : List<String>
publicationDate: Date
isOfType
Publication Type
identifier
type
isAuthor
hasDocument
Author
forename
surname
Journal
journal
hasAffiliation
Affiliation
affiliation

## Definition 7.4 (Social Network)

A *Social Network* is a Graph $G = (V, E)$ with vertices (nodes) $v \in V$ and edges (relations) $e \in E$. Both edges and vertices are part of previously well-defined categories, $V \subseteq C_1 \cup C_2 \cup ... \cup C_n$ and $E \subseteq R_1 \cup R_2 \cup ... \cup R_m$ The network $G$ may either be directed or undirected.

- We can provide a formal definition of a social network (see previous chapter) that shows both concepts are equivalent.
- Considering Knowledge Graphs as the framework to represent data opens the complete graph-theoretical toolbox to data science.
- However, (semantic) questions need to be 'translated' into graph-theoretical questions (often called 'graph queries').

- In general, we need to transform real-world questions into a ‘graph query’.
- Graph databases and build-in functions are not very scalable.
- A distributed and federated system might be a solution.
- Two examples:

> *(Question 4) What is the shortest way between {Entity1} and {Entity2} and what is on that way?*
> *(Question 12) How far apart are {document1} and {document2}?*

(Q4)
```
match (entity1:Entity {preferredLabel: "axonal transport"}),
(entity2:Entity {preferredLabel: "LRP3"}) call
algo.shortestPath.stream(entity1, entity2) yield nodeId return
algo.asNode(nodeId)
```

# Optimization of Retrieval Algorithms on Knowledge Graphs

- Converting semantic questions into ‘graph queries’ helps to understand the algorithmic challenge and complexity of a problem.
- More examples:

  (2) *Which genes play a role in two diseases?* (RPQ)

  (9) *Are there authors within the same affiliation who make contradictory statements regarding protein Entity1 and protein Entity2?* (CRPQ)

  (10) *Do the data in the literature correlate with the concomitant diseases for illness Entity1? So are the genes mentioned in Entity1 documents also mentioned in Entity2 documents of the concomitant disease?* (CRPQ)
- Conjunctive Queries (CQ), Regular Path Query (RPQ), Conjunctive Regular Path Queries (CRPQ), Extended Conjunctive Regular Path Queries (ECRPQ)

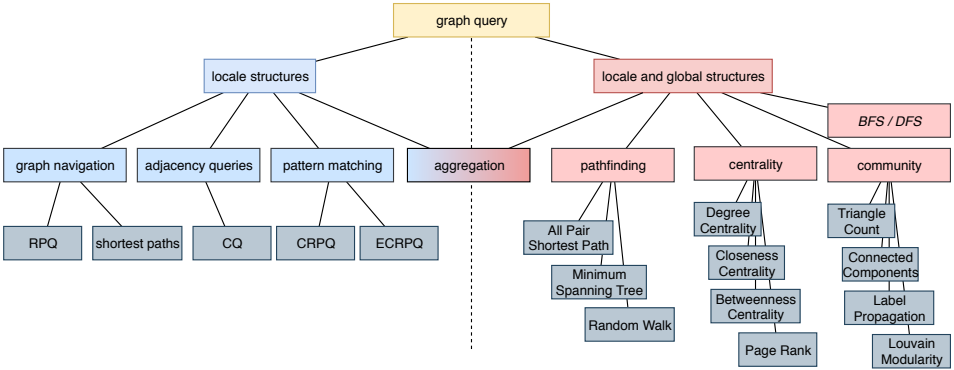


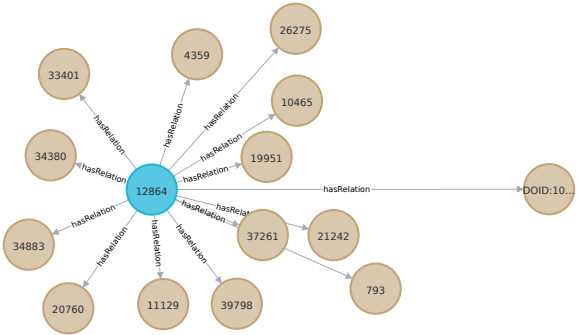

Figure 16: An overview of the categories for graph queries unified from literature sources. These categories give a first overwiev and a categorization scheme for graph queries and their complexity.

- The schema divides the categories for graph queries into *local structures* and *local & global* structures.
- The second structure category is called *local & global*, because some of the graph algorithms can act locally by specifying a start node or a subgraph.
- Furthermore, some categories, such as *CRPQ* or *ECRPQ*, were identified as subcategory of other categories. This is illustrated by the hierarchical structure of the categorization scheme.
- The category *Aggregation* belongs to graph queries that search for both local and global structures. For example, the category *Aggregation* can include questions such as “What is the degree of node A?” or “What is the average of the graph?”, the former referring to local and the latter to global structures.

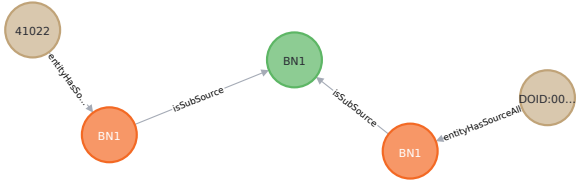
Search for graph patterns:

- The evaluation problem for this query is $\mathcal{NP}$-complete.
- Since the size of the query is in practice much smaller than the size of the dataset, the term *Data Complexity* is used, where the query is assumed to be fixed and the input depends only on the graph.
- In contrast, the more general *Combined Complexity*, which considers a variable query and a variable graph database, is used.
- Using data complexity, graph pattern queries can be evaluated in polynomial time.
- Data complexity is often considered a more relevant measure, since graphs are often very large, whereas queries are very small.

- While some problems are known to have solutions running in polynomial time (like pathfinding), others are well known to be quite hard.
- For example graph navigation and pattern matching are more complex.
- RPQ is in $\mathcal{P}$, CRPQ and ECRPQ are $\mathcal{NP}$-complete.
- Degree Centrality: $O(|V|^2)$
- Betweenness Centrality: $O(|V|^3)$
- Shortest Path: $O(|V| \cdot log(|V| + |E|)$
- Group Closeness Maximization (GCM) or Maximum Betweenness Centrality are $\mathcal{NP}$-hard.




Example for ECRPQ.




Example for shortest path.

### Definition 7.5 ($k$-Clique Problem)

Let $G = (V, E)$ be a graph an $k \in \mathbb{N}$. We want to obtain a clique with $k$ vertices, if one exists, or a special value indicating that there is no $k$-clique otherwise.

### Definition 7.6 (Subgraph Problem)

Let $G$ and $H$ be two graphs. Exists a subgraph $G' \subseteq G$ with $G' \simeq H$?

- $k$-Clique is $\mathcal{NP}$-complete, thus also Subgraph.
- We may try to solve it with Brute-Force Algorithms or Backtracking.

- It is possible to store and analyze longitudinal data in knowledge graphs.
- This is an important topic in medical informatics, for example when working with longitudinal patient records.
- Although research started early on versioning RDF knowledge bases, only little research has been done on this field.
- Some attention was paid to the field of the evolution of data structures within information management, and the decentralized collaborative work on knowledge resources.
- Other researchers were interested in parallel world frameworks to analyze scenarios in knowledge graphs.
- Nevertheless, a generic framework for modeling longitudinal data in knowledge graphs is still missing.

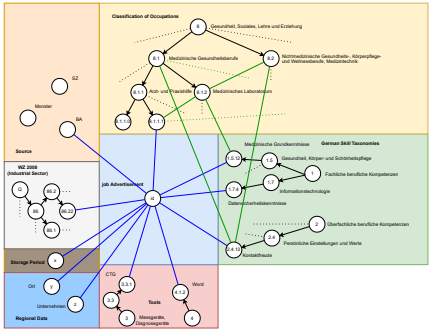



Figure 17: Knowledge graph representation of German job ads. Different layers within the knowledge graph have a different background color, edges from existing data structures and ontologies are black, blue nodes are either extracted from ads using text mining or structural information. Green edges are obtained by external data sources, for example from the German Federal Employment Agency (Bundesagentur für Arbeit - BA).

- We will provide an example usecase focusing on an example use case from computational social sciences.
- In labor market research, extracting skill requirements from job advertisements (short: job ads) becomes a feasible approach to observe which skills are in demand by employers.
- Job ads are one way for a company to recruit new employees. Beside general information about the hiring company and the working conditions, they document the current skill needs on the labor market. By that, they can be a good indicator of whether training programs need to be adjusted.
- Since skills can be expressed in various ways, one challenge for skill extraction is to group these expressions into skill concepts.

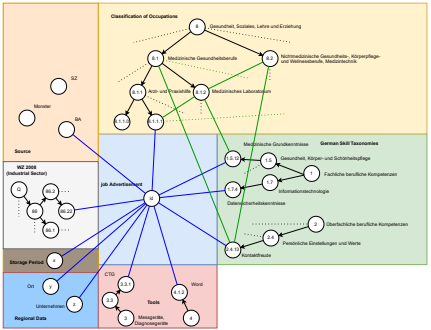



Figure 18: Knowledge graph representation of German job ads. Different layers within the knowledge graph have a different background color, edges from existing data structures and ontologies are black, blue nodes are either extracted from ads using text mining or structural information. Green edges are obtained by external data sources, for example from the German Federal Employment Agency (Bundesagentur für Arbeit - BA).

- In addition, for a longitudinal view on how skill demands develop, it is necessary to build a model which is capable of not only utilizing these data within a knowledge graph with contextual data, but also for efficient analysis. We present a detailed overview of the knowledge graph representation in Figure 18.
- The main research question of this example related to network dynamics: How can we model longitudinal knowledge graphs on job ads for an efficient analysis of the development of skills while preserving all contextual data?

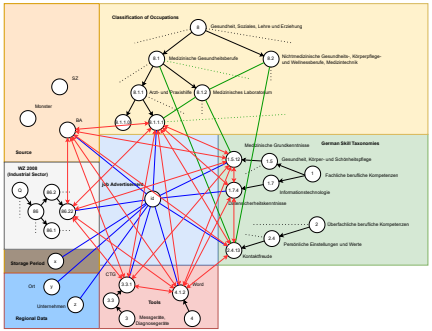



Figure 19: Knowledge graph representation of German job ads. In this case the red edges build the pseudo-triangles explaining the context of the job ad.

| | Content | Structure |
|---|---|---|
| $E_1$ | Job Ads | unstructured |
| $E_2$ | Classification of Occupations | Taxonomy |
| $E_3$ | Sources | unstructured |
| $E_4$ | German Skill Taxonomy (AMS) | Taxonomy |
| $E_5$ | Tools | Taxonomy |
| $E_6$ | Industrial Sectors | Taxonomy |
| $E_7$ | Storage Period | unstructured |
| $E_8$ | Regional Data | unstructured |

Table 2: Different layers within the knowledge graphs of German job ads.

In general, for a node $n \in V$, the neighborhood $N(n)$ contains all relevant contextual information. But usually information is best understood using information-triangles. Thus every two nodes $v, w \in N(n)$ form an implicit triangle $v, n, w$ and when adding an additional edge $(v, w)$ this forms a triangle $K_3$.

### Definition 7.7 (Pseudo-Triangle)

Let $G = (E, R)$ be a knowledge graph and let $n, v, w \in G$ be three nodes in $G$. Moreover, let $n \in E_i$ for some $i$ and let $v, w \in N(n)$. Then $n, v, w$ form a pseudo-triangle in $G$.

- It is quite obvious that this general construction of contextual information allows a better interpretation of data.
- For example, the skill ‘basic knowledge in medicine’ (‘Medizinische Grundkenntnisse’) has a different meaning in the context of a doctor’s assistance (‘Arzt- und Praxishilfe’) than in the context of a person working in medical informatics.
- The same holds for particular tools like word-processors (‘Word’) for an assistant or a journalist.
- This information can be crucial for analyzing the ads and also for training models for text mining and natural language processing.

It appears that in this example we are working on a knowledge graph $G = (V, E)$ with eight different layers, thus

$$G = E_1 \cup E_2 \cup ... \cup E_8 \cup T_1 \cup I_1 \tag{14}$$

- Starting with Equation 14 we can extend the knowledge graph with the information for one particular time point $t$, in our case for a year:

$$G^t = E_1^t \cup E_2^t \cup ... \cup E_8^t \cup T_1^t \cup I_1^t \tag{15}$$

- With this, we can build a generic graph model comprising multiple times $T = \{t_1, ..., t_m\}$:

$$G_T = G^{t_1} \cup G^{t_2} \cup C_{t_1,t_2} \cup G^{t_2} \cup G^{t_3} \cup C_{t_2,t_3} \cup ... \\ G^{t_{m-1}} \cup G^{t_m} \cup C_{t_{m-1},t_m} \tag{16}$$

- In this case, $C_{t_i,t_k}$ comprises all edge relations from $G^{t_i}$ to $G^{t_j}$.
- This contains relations like `isEqual` if two entities are equivalent or `isSuccessor` if an entity in $G^{t_j}$ is the successor of an deprecated element in $G^{t_i}$. See Figure 20 for an illustration.

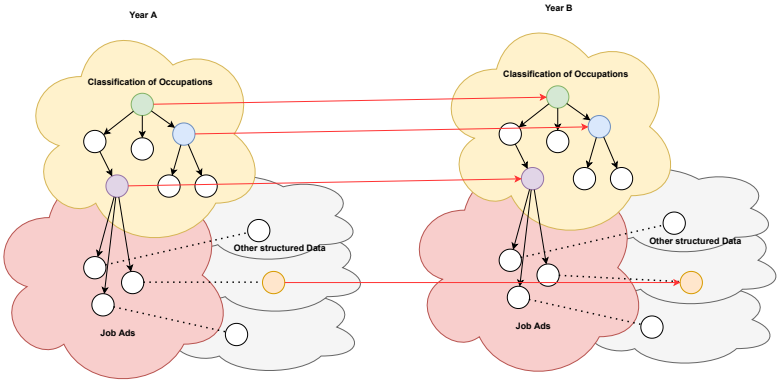



Figure 20: Illustration of a subset of the knowledge graph for two years $A$ and $B$, $G^A$ and $G^B$. The red nodes are in $C_{A,B}$ and describe the relations between changed or continues elements in both years.

- Thus, a first step to shrink the volume of the knowledge graph is to merge the multiple existence of elements in multiple years.
- Thus, we search for maximal paths
$$P = p_1, ..., p_m$$
- in $G_T$ where $p_i \in E_j \; \forall p_i \in P$.
- The edges between $p_i$ and $p_{i+1}$ are either `isEqual` or `isSuccessor` edges, see Figure 21.

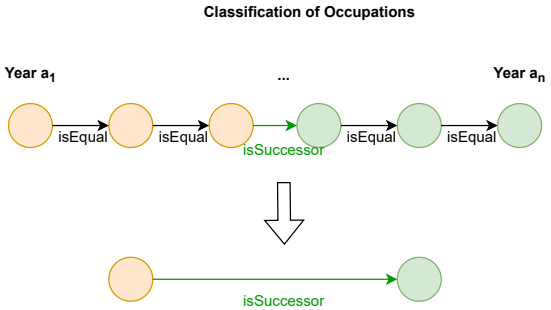



Figure 21: Merging maximal paths $P = p_1, ..., p_m$ containing equal data in multiple years.

- Thus for every edge $(p_i, p_{i+1}) \in C_{t_j,t_k}$ we can either merge $p_i$ and $p_{i+1}$ if they are the same (`isEqual`) or leave the `isSuccessor` edges. In our case we are in particular working on $E_2$, the classification of occupations.
- This can be done with depth-first search, see Algorithm 13, because we explicitly only use the directed subgraph induced by

$$R = G_T\left[C_{t_1,t_2} \cup C_{t_2,t_3} \cup ... \cup C_{t_{m-1},t_m}\right]$$

- The worst-case behavior is in $O(E(R) + V(R))$ and since every node $p_i$ has at most $\Delta(E_2)$ neighbors in $E_2$ and at most $N(E_1)$ neighbors in $E_1$ the time complexity of merging the nodes is $O(\Delta(E_2) + N(E_1))$. Thus, the runtime of this step is linear, $O(n)$ in $G_T$. We denote the graph after step 1 with $G_T^1$.

**Algorithm 13** STEP-1

**Require:** Knowledge Graph $G_T$ with layer (tree) $E_x^t$ and mappings $C_{t_i,t_{i+1}}$ for all $t \in T = \{t_1, ..., t_m\}$
**Ensure:** Shrinked $G_T^1$

$M = N(\bigcup_{i=1,...,m} E_x^i) \cup E(\bigcup_{i=1,...,m-1} C_{t_i,t_{i+1}})$
$V = N(\bigcup_{i=1,...,m} E_x^i)$
$P = []$
**for** every $v \in E_x^i$ with $v \in V \; \forall i \in \{1, ..., m\}$ **do**
  $p =$DFS$(M, v)$
  del$(V, y) \; \forall y \in p$
  $P$.add $(p)$
**return** $G_T^1 = \bigcup_{i=1,...,len(P)} (p_i \in P)$

- In a next step, we can merge all job ads for a given year preserving the further links to other structured data.
- Thus for every time point $t$ and every $v^t \in E_2^t$ we merge all nodes in $N = \{n^t |\ n^t \in N(v^t)$ and $n^t \in E_1^t\}$ to a meta node $a_t$ and add an edge $(v^t, a_t)$ with weight $|N|$, see Figure 22.
- These form pseudo-triangles in $E_1$.

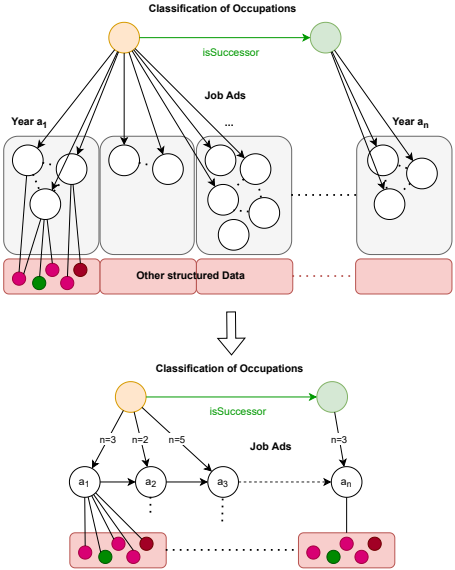



Figure 22: Merging all job ads for a given year preserving the further links to other structured data.

The runtime of this step is in $O(tN(E_1)N(E_2)$ and thus is quadratic, $O(n^2)$ in $G_T$, see Algorithm 14. We denote the changed graph after step 2 with $G_T^2$ and the new shrinked nodes in $E_1$ with $E_1^2$.

**Algorithm 14** STEP-2

**Require:** Knowledge Graph $G_T$ with an unstructured layer $E_x^t$ and a target layer $E_y^t$ containing paths in $P$ and mappings $C_{t_i,t_{i+1}}$ for all $t \in T = \{t_1, ..., t_m\}$ and
**Ensure:** Shrinked $G_T^2$
$V = \emptyset$
$E'_x = \emptyset \forall t \in \{1, ..., m\}$
$P = []$
**for** every $p \in P$ in $E_y^t$ **do**
  **for** every $p_i \in P = \{p_1, ..., p_z\}$ **do**
    **for** every $t \in \{1, ..., m\}$ **do**
      $V' = N(p_i) \cap E_x^t$
      add $v'$ to $E_x^t$
      change all edges $(\hat{v}, u) \forall \hat{(v)} \in V'$ and $u \notin E_x$, $u \notin E_y$ to $(v', u)$
**return** $G_T^2 = (G_T \setminus E_x) \cup E'_x$

- Before continuing with a possible third step of shrinking graph structures, we should consider the theoretical results.
- Given the question, how the description of skills in job ads evolves in job classifications over the years, in the initial graph $G_T$ we need to consider the following steps:
- Consider the evolution of any classifications $v$ for all times $T = \{t_1, ..., t_m\}$, runtime $O(m \max_i V(E_2^i))$.
- Consider all skills for any job ad in $N(v^t)$ for all times, runtime $O(V(E_1^t)V(E_4^t))$.

**Algorithm 15** APPROACH ON $G_T$

**Require:** Knowledge Graph $G_T$ with two structured layers $E_s$, $E_i$ and an unstructured layer $E_d$
**Ensure:** Set $S_t \subset E_s$ for all $t = \{t_1, ..., t_m\}$

$S = []$
**for** every $t \in \{1, ..., m\}$ **do**
  **for** every $v \in E_i^t$ **do**
    **for** every $u \in N(v) \cap E_d$ **do**
      $S[t, v] = N(u) \cap E_s$
**return** $S$

Thus, the average runtime is in $O(n^3)$, see Algorithm 15.

For the graph with shrinked pseudo-triangles this reduces to linear runtime:

- Consider any shrinked path $p$ in $E_2^1$, runtime $O(\max_i N(E_2^i))$.
- Consider all skills for all times, runtime $O(m)$.

See Algorithm 16 for pseudocode.

**Algorithm 16** Approach on $G_T^2$

**Require:** Knowledge Graph $G_T^2$ with two structured layers $E_s$ with shrinked paths, $E_i$ and an unstructured layer $E_d$ with shrinked pseudo-triangles

**Ensure:** Set $S_t \subset E_s$ for all $t = \{t_1, ..., t_m\}$

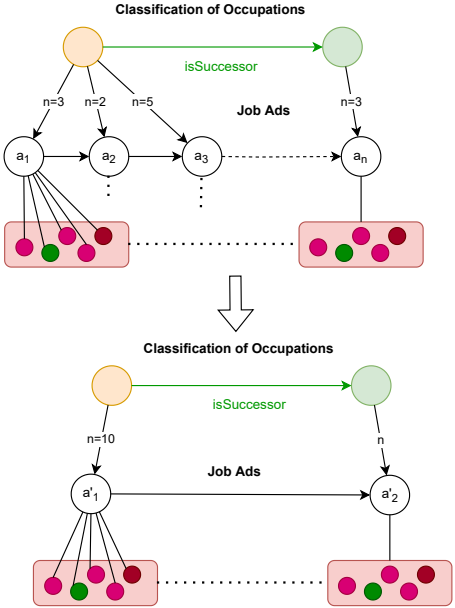
$S = []$

2: **for** every $p \in E_s$ **do**

$S[p_i, v] = N(N(p_i \cap E_d) \cap E_s) \; \forall p_i \in p$

4: **return** $S$

- We could continue the process of shrinking (pseudo-)triangles because as we can see in Figure 22 new triangles have been build between $E_1^2$ and $E_2^2$.
- This would result in merging all job ads for a given job classidication preserving the further links to other structured data, see Figure 23.


Classification of Occupations
isSuccessor
n=3
n=2
n=5
Job Ads
n=3
$a_1$
$a_2$
$a_3$
$a_n$
Classification of Occupations
isSuccessor
n=10
n
Job Ads
$a'_1$
$a'_2$

- With this third step we can reduce the data complexity, but while in step 1 we do not lose any relevant data,
- in step 2 we lose the information about specific job ads while preserving the information for a complete time set.
- In step 3 we lose the information about those years with a constant job classification.

Thus we can make the following observations:

- Structured data like taxonomies and ontologies can be shrinked without any data loss.
- Shrinking unstructured data in triangles or pseudo-triangles always goes along with data loss of particular data points while accumulated information might be preserved.
- Thus, it highly depends on the initial research question which layers and information can be shrinked to improve the runtime of algorithms.

For the given research question, steps 1-2 are the maximum reduction of the initial graph if considering the change for years. Step 3 would be an additional optimization of the initial graph when considering the research question with respect to the change in the classification of occupations.

Graph data is too complex: existing machine learning algorithms usually fail.

- They are specialized in simple data types (images with fixed structure and size, text and speech as sequences).
- In graphs, data points are not independent (edges) and the data structure is not fixed (node degree).

- Graph Neural Networks (GNNs).
- So-called ‘graph embeddings’.
- Learning graph structures.
- Link prediction.

- Link prediction on graphs, for example on knowledge graphs and social networks, is usually done using embeddings which form a low-dimensional representation of the graph. The main assumption is that they provide an accurate reconstruct of the graph.
- In general, factorisation, random walk and deep learning approaches are used.
- While these approaches have also been considered for applications in other domains, a general approach towards learning links solely based on graph structures is yet missing.
- Some researchers have tried to propose features based on graph structures and found promising results based on a large amount of features modeling different aspects of the graph structure.

Scores based on the topology of the graph

- Link prediction belongs to the field of computational analysis of a network, where the nodes represent persons or entities and the edges represent relations.
- These networks are dynamic and change over time.
- The link prediction problem deals with a section of such a network at a time $t_0$ and asks for the most accurate predictions possible for edges that do not yet exist at time $t_0$ and will be added at a later time $t$.
- Among other things, the network's own topology plays a crucial role. To be able to quantify this topology different neighbourhood measures from graph theory and their relative effectiveness are investigated.

Liben (2008) proposes a so-called score is for the measure of this effectiveness. It is calculated in different ways. Examples are:

- Common Neighbours: Given a graph $G = (V, E)$, $Score(x, y) := |\Gamma(x) \cap \Gamma(y)|$ describes the number of common neighbours of two nodes $x, y \in V$. Here, $\Gamma(v)$ denotes the direct neighbourhood of a node $v \in V$.
- Preferential Attachment: Given again a graph $G = (V, E)$. The underlying premise is the assumption that the probability that a new edge contains the node $x \in V$ is proportional to $|\Gamma(x)|$. Since the measure was originally conceived for predicting future collaborations between two authors, this yields $Score(x, y) := |\Gamma(x)| \cdot |\Gamma(y)|$. This builds on the idea that nodes with many edges have a higher probability of even more edges.

- Adamic/Adar: The coefficient found here originally yields a measure that two homepages are strongly connected. For this purpose, features $z$ are computed from a feature base set $F$ of the two nodes, here web pages, and the commonality is defined as:

$$\sum_{z:\text{features shared by } x,y} \frac{1}{\log(\mathit{frequency}(z))}$$

This gives less weight to more frequent features than to less frequent ones. If features are to be left out and only the topology of the graph is to be considered, the following score is used for two nodes $x, y \in V$ of a graph $G = (V, E)$:

$$\mathit{Score}(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log(\Gamma(z))}$$

These measures belong to methods based on node adjacency.

- First, a simple Markov chain of order $n$ is considered.
- The idea is to be able to calculate the probability of future states occurring.
- The order indicates on how many previous states the next one depends.
- In a first-order Markov process, the next state depends only on the current state. At the beginning, the system is in the initial state.

**Definition 7.8**

A Markov process is understood to be a tuple $(S, A, \delta)$. Here $S$ describes the finite set of states, $A$ the set of possible actions, and $\delta$ the state transition function.

For each pair $(s_t, a_t)$ with $s_t \in S, a_t \in A$ the state $s_t$ transitions via $\delta(s_t, a_t)$ to the state $s_{t+1}$. The transitions in this case are usually given in probabilities. The choice of action depends on the current state and can be represented as a function $\pi : S \to A; \pi(s_t) = a_t$ . It is also called a strategy.

- Hidden Markov models are used to represent probability distributions over sequences of observations.
- A distinction is made between the observation $X_t$ and the state $Z_t$ at time $t$.
- The latter is hidden, hence the name of the model.
- Here, as in the 1-step Markov chains, the so-called Markov property is assumed: $Z_t$ at time $t$ depends only on $Z_{t-1}$ at time $t-1$.
- An example of this can be seen in Figure 24.
- The time $t$ need not be an explicit time and can also be implicitly considered as a location within the sequence.
- The overall probability distribution of a sequence of states and observations can be expressed as an equation as follows:

$$P(Z_{1:N}, X_{1:N}) = P(Z_1)P(X_1|Z_1)\prod_{t=2}^{N} P(Z_t|Z_{t-1})P(X_t|Z_t)$$

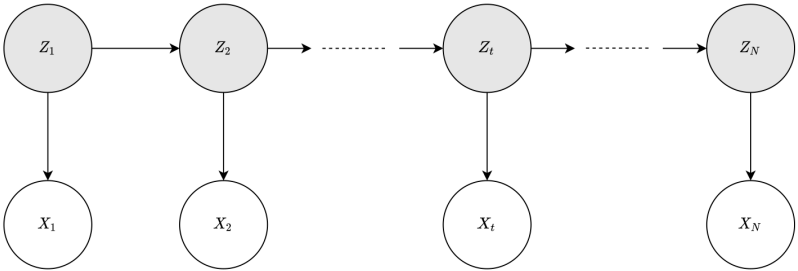



Figure 24: Example of a hidden Markov model: $Z_t$ describes the state and $X_t$ the observation dependent on it at time $t$.

- Since the states are hidden and only the observations are considered, which in turn depend on the states, the probability of an N-element sequence is represented by a product of conditional probabilities. Moreover, except for the initial state, each state depends on the previous one.

There are five elements that characterise a hidden Markov model:

- the number $K$ of states that can be assumed in the model. The states are represented as $K \times 1$ vectors with binary values such that the $k$-th state at time $t$ takes the value 1 in the $k$-th row and 0 everywhere else.
- the number $\Omega$ of distinct observations that can be observed in the model. Analogous to the states, an $\Omega \times 1$ vector is used.
- the state transition model $A$: This is also called the state transition probability distribution and describes the probability of changing from a state $Z_{t-1,i}$ to a state $Z_{t,j}$ within one time step. Here $i, j \in 1, ..., K$. This can be formulated as follows:

$$A_{i,j} = P(Z_{t,j} = 1 | Z_{t-1,i} = 1)$$

  Each row of $A$ sums up to 1 in this case.
- the observation model $B$ is an $\Omega \times K$ matrix whose elements $B_{j,k}$ give the probability of making the observation $X_{t,k}$ given the state $Z_{t,j}$:

$$B_{j,k} = P(X_t = k | Z_t = j)$$

- the initial state distribution $\pi$ is a $K \times 1$ vector with $\pi_i = P(Z_{1,i=1})$.

The model is often abbreviated in literature as $\lambda = (A, B, \pi)$.

## Markov Random Fields

- Let $G = (V, E)$ be an undirected graph.
- The nodes $v \in V$ correspond to the random variables which can assume the states.
- Here, these depend only on the states of the random variables $u$ of their Markov cover $B_v := \{u : (v, u) \in E\}$.
- This is expressed in the following equation:

$$P(x_1, ..., x_n) = \frac{1}{Z} \prod_{c \in C} F_c(x_c)$$

- Here $C$ is the set of maximal cliques of the graph.
- The functions $F$ are non-negative and depend on the variables within a clique $c$.
- For normalisation, a function $Z = \sum_{x_1,...,x_n} \prod_{c \in C} F_c(x_c)$ is used so that the distribution sums up to 1 overall.

Conditional Random Fields are a special case of Markov Random Fields and belong to the field of supervised learning. Instead of only considering the probability for a label sequence $y$, here the probability of a label sequence $y$, conditioned by an observation sequence $x$, is determined:

$$P(y|x) = \frac{1}{Z(x)} \prod_{c \in C} F_c(x_c, y_c),$$

$$Z(x) = \sum_{y \in Y} \prod c \in CF_c(x_c, y_c)$$

The normalisation function $Z(x)$ now also depends on $x$.

In other literature, the definition of a (linear chain) Conditional Random Field is the conditional probability

$$p(y_{1:n}|x_{1,n}) = \frac{1}{Z} \exp\left(\sum_{n=1}^{N}\sum_{i=1}^{F} \lambda_i f_i\left(y_{n-1}, z_n, x_{1:N}, n\right)\right).$$

- Within the exponential function, the first sum is over $n = 1, ..., N$, which indicates the position of a word, or here a node, within the sequence.
- The second sum iterates the features $f_i$ weighted by the scalars $\lambda_i$, $i = 1, ..., F$.
- The values for the weights must be given or learned by the CRF model.
- They ensure that certain labels are preferred or even avoided.
- For a given sequence, several features can be active at the same time, i.e., not equal to 0.
- This is called overlapping features.
- This can happen because, unlike in hidden Markov models, it is also possible to look at subsequent or previous elements of the sequence.

- To train, fully labelled training sequences $(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})$ are required, where $x^{(i)} = x^{(i)}_{1:N_i} \ \forall i \in 1, ..., m$.
- Thus, the conditional probability of the training data is maximised:

$$\sum_{j=1}^{m} \log p\left(y^{(j)} | x^{(j)}\right)$$

- This is computed by default employing algorithms that use the gradient descent method.

To assess the quality of the prediction, the F1-score (also balanced F-score or F-measure) is used. This can be regarded as a weighted average of the precision and the recall. The best value is 1 and the worst value is 0. The formulas used for this are:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}, \tag{17}$$

$$Precision = \frac{TPR}{APR}, \tag{18}$$

and:

$$Recall = \frac{TPR}{APS}. \tag{19}$$

Here *TPR* means true positive results, *APR* means all positive results and *APS* are all samples that should have been identified as positive.

- Link prediction is used to predict possible, initially non-existent edges for the previously constructed graph.
- A path $p$ is considered first.
- Then, for each node $v$ contained in $p$, all neighbouring nodes $u \in \Gamma(v)$ are taken as possible labels.
- Thus, each node can be used both as a node of a path and as a label for other nodes, see Figure 25.

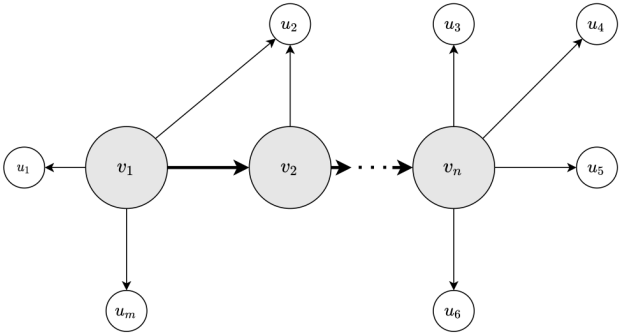



Figure 25: The input path $p$ consists of the nodes $v_i$, $i = 1, ..., n$ highlighted in grey. The white nodes $u_j$, $j = 1, ..., m$ serve as labels of the nodes of $p$.

Creating one-node paths

- The simplest form offers a path of length one, i.e. a single node and its direct neighbourhood.
- It is specified which node type is considered, e.g. patients or images in clinical data.
- Then we need to find the direct neighbourhood $\Gamma(v)$ of these nodes $v \in G$ and $\Gamma$ is stored as a set of labels $l(v)$ for $v$.
- Since CRF can only assign one label to each node v at a time, criteria must be used for selection.
- For this purpose, scores can be used to select nodes with, for example, the highest score.
- We might also consider alphabetical sorting.
- The choice of the method for probing the labels on the one hand influences the result and on the other hand also the runtime.

## Example 7.9

(Q1) `MATCH (p:Patient)-[]-(a) RETURN p.nodeUID as patientNode, a.nodeUID as labelNode, gds.alpha.linkprediction.commonNeighbors(p,a) AS score ORDER BY p.nodeUID,score DESC,a.nodeUID`

(Q2) `MATCH (p:Patient)-[]-(a) RETURN p.nodeUID as patientNode, a.nodeUID as labelNode, gds.alpha.linkprediction.totalNeighbors(p,a) AS score ORDER BY p.nodeUID,score,a.nodeUID`

(Q4) `MATCH (p:General_Image)-[]-(a) RETURN p.nodeUID as imageNode, a.nodeUID as labelNode ORDER BY p.nodeUID,a.nodeUID`

(Q5) `MATCH (p:Date)-[]-(a) RETURN p.nodeUID as dateNode, a.nodeUID as labelNode ORDER BY p.nodeUID,a.nodeUID`

## Example 7.10

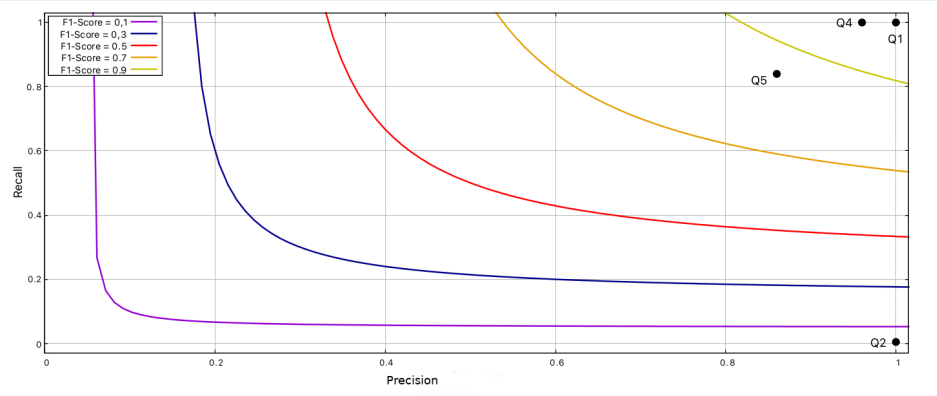
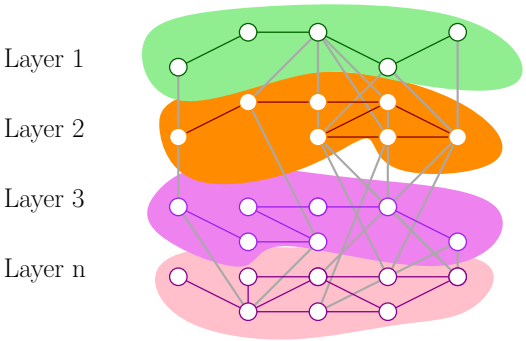



Figure 26: Precision recall diagram for queries Q1 - Q5.

# Knowledge Discovery using Semantic Graph Embeddings




With the neighborhood $N(E_i)$ every node set $E_i \in \{E_1, ..., E_n\}$ induces a subgraph $G[E_i] \subset G$:

**Definition 7.11 (Semantic Graph Embeddings)**

With $G^c[E_i] = G[E_i] \cup N(E_i)$ we denote the extended context subgraph or semantic graph embedding which also contains the neighbors of each node in $G$, which is context of that node. With $G^c_L[E_i] = G^c[E_i] \cap L$ we denote the graph embedding on layer $L \subset G$.

We set $\mathcal{E}(N) = G^c[N]$ and $\mathcal{E}_L(N) = G^c_L[N]$.

# Knowledge Discovery using Semantic Graph Embeddings

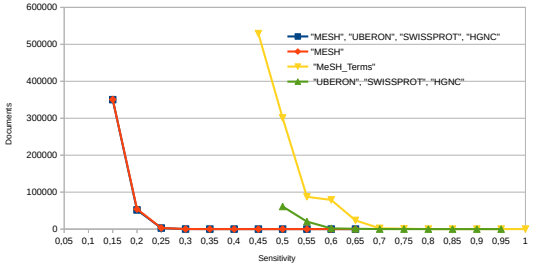



Figure 27: Example outputs of heuristic for Corpus "Alzheimer Disease" with different layers.

**Algorithm 17** *s*-GRAPH-DESCRIPTION

**Require:** $N = \{n_1, ..., n_n\} \subset L$ and descriptive elements $con(n_i) = \{c_1, ..., c_m\} \subset V$, maxiter as maximum of iterations, $s$ as sensitivity

**Ensure:** A semantic graph description $\mathcal{E}(N) = (V', E')$ of $N$ with elements in $V \cap L$.

$con' = con$

2: **for** every $v \in N$ **do**

**while** iteration<maxiter AND $con'(v) > (s \cdot con(v))$ **do**

4: remove $c \in con'(v)$ with maximum weight

**return** $Z = \vee_{v \in N}(\wedge_{x \in con'(v)})$

# Knowledge Discovery using Semantic Graph Embeddings

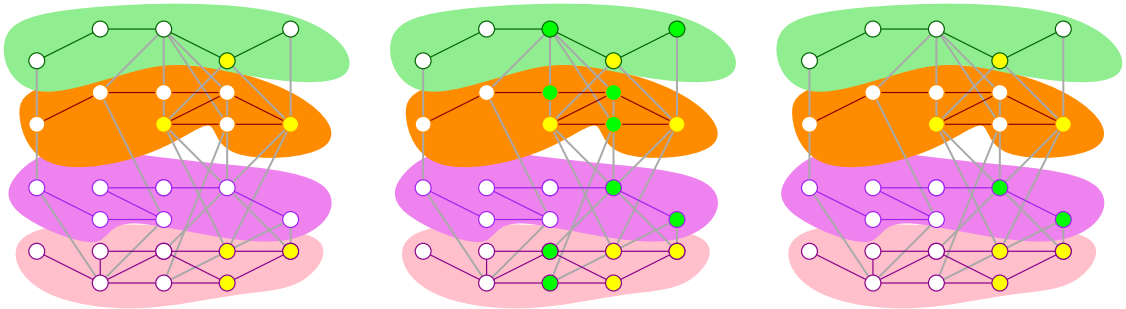
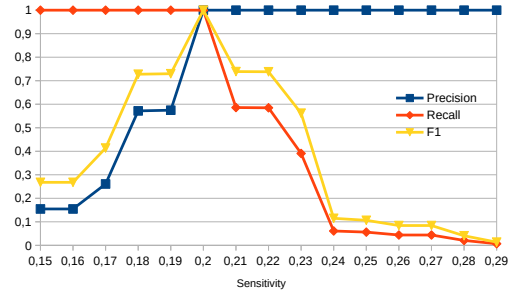
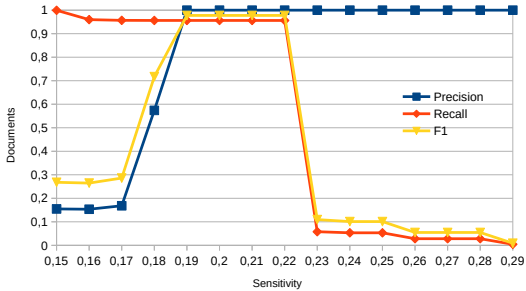


Figure 28: Illustration of the steps for generating a semantic graph embedding $\mathcal{E}_L$ where $N$ is given by the yellow nodes and $L$ is given by the pink layer (1). Subfigure (2) depicts the output of algorithm, $\mathcal{E}(N) = N \cup con(N)$. Limiting this to $L$ returns $\mathcal{E}_L = \mathcal{E}(N) \cap L$, see subfigure (3).

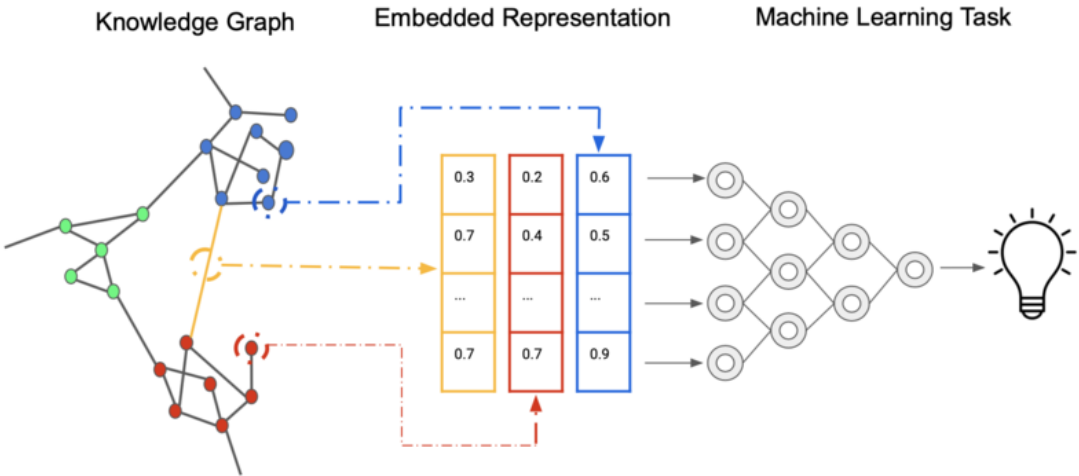
# Knowledge Discovery using Semantic Graph Embeddings




- Context is the glasses to view the world. Do we really see reality?

- A *graph embedding* ('Knowledge Graph embedding', KGE) translates each node and edge of a graph$G$ into a vector of a given dimension $d$, called embedding dimension.



To enhance the embedding data in the *representation space*, we may use the following methods:

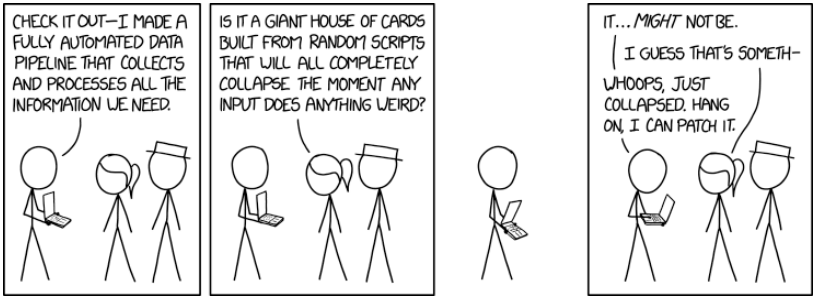
- *Scoring function* to evaluate the performance of representation.
- Provide *encoding models* for the interaction between the embedded representation nodes and edges.
- Any other *additional information* may be used.

Different models and technologies are used:

- Tensor decomposition models
- Geometric models
- Deep learning models (e.g. Convolutional neural networks)

# AI and Graphs

- **Context information are key for understanding data**.
- Knowledge Graphs (or: the idea of a KG or: linked data) are a way to make data **findable**, **accessible**, **interoperable** and **re-usable**.
- This will help to push the frontiers of several data mining challenges.




Source: https://xkcd.com/2054/.

# FAIR Data


**F**indable, **A**ccessible, **I**nteroperable, **R**e-usable.

- Web → Semantic Web
- Around 2012: 5 Stars Open Data (Tim Berners-Lee)
- DKAN, WikiData, ...

Source: https://www.w3.org/DesignIssues/LinkedData.html

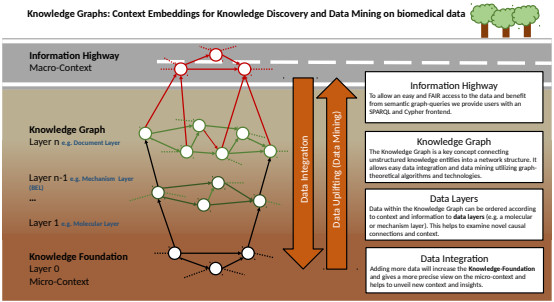
# Linked Data and Knowledge Graphs

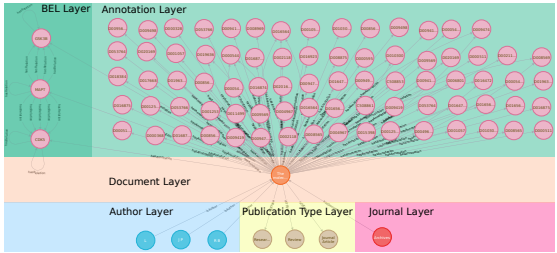
- **Data linking** is usually just about content and a few metadata (`X and Y and [author:Z]`).
- **Data-centric** vs. **knowledge-centric** perspective.
- What exactly is a knowledge graph? Different perspectives from Engineering, Data Architecture, Data Engineering.

| **RDF** Resource Description Framework | **LPGs** Labeled Property Graphs |
|---|---|
| Triple Stores | Graph Databases |
| Query a graph (SPARQL) | Traverse a graph (e.g. Cypher) |
| Vertices and edges have **no internal structure** | Vertices and edges have **an internal structure** |
| RDF data rules + Ontologies | |

# Implementation – Data Schema

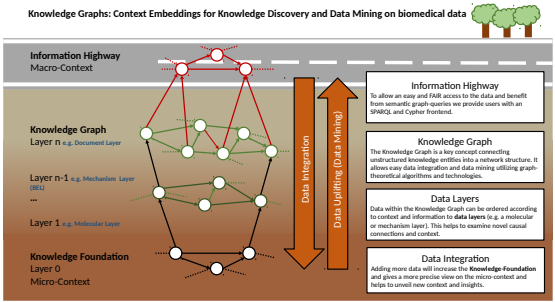
- **Knowledge representation** both as general as possible and as specific as necessary.

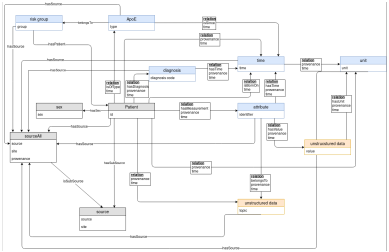


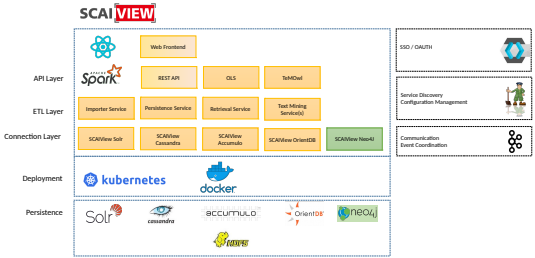
# Implementation – Data Schema

- **Knowledge representation** both as general as possible and as specific as necessary.



# Implementation – Infrastructure

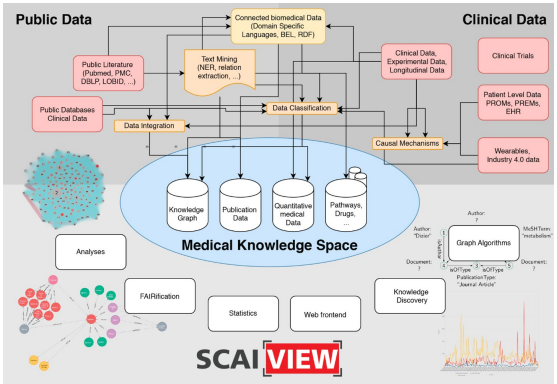



- Integration in a scalable microservice architecture?
- Java using Spring Boot and Spring Data or Python.
- **Graph Databases**: Neo4j, OrientDB

## Example 7.12

- **SCAIView**: Document, Text Mining (NER and RE)
- **BiKMi**, **OLS**, BEL networks.

# Implementation – Infrastructure




- Integration in a scalable microservice architecture?
- Java using Spring Boot and Spring Data or Python.
- **Graph Databases**: Neo4j, OrientDB

## Example 7.12

- **SCAIView**: Document, Text Mining (NER and RE)
- **BiKMi**, **OLS**, BEL networks.

# Optimization of Retrieval Algorithms on Knowledge Graphs

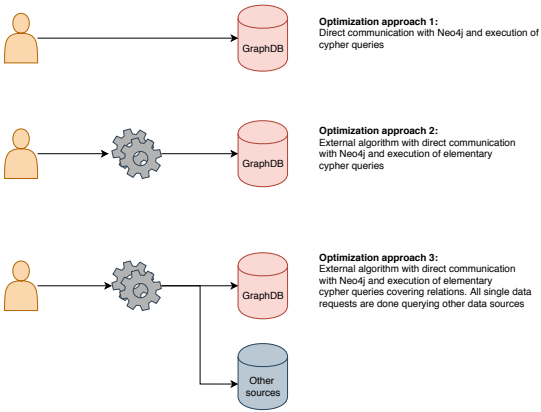



Figure 29: An overview of the optimization approaches discussed in this paper. The first approach contains the basic Cypher query, the second approach transfers the algorithm to a different system. The third approach relies on a polyglot persistence architecture and excludes all time-consuming queries that can be answered by a key-value store.

# Optimization of Retrieval Algorithms on Knowledge Graphs

Queries 4 and 12 are both a typical shortest path problem:

(Q4) *What is the shortest way between* {*Entity1*} *and* {*Entity2*} *and what is on that way?*

(Q12) *How far apart are* {*document1*} *and* {*document2*}*?* Thus both problems can be solved using Cypher:

(Q4)
```
match (entity1:Entity {preferredLabel: "axonal transport"}), (entity2:Entity {preferredLabel: "LRP3"}) call algo.shortestPath.stream(entity1, entity2) yield nodeId return algo.asNode(nodeId)
```

(Q12)
```
match (doc1:Document {documentID: "PMID:16160056"}), (doc2:Document {documentID: "PMID:16160050"}) call algo.shortestPath.stream(doc1,doc2) yield nodeId return algo.asNode(nodeId)
```

# Optimization of Retrieval Algorithms on Knowledge Graphs

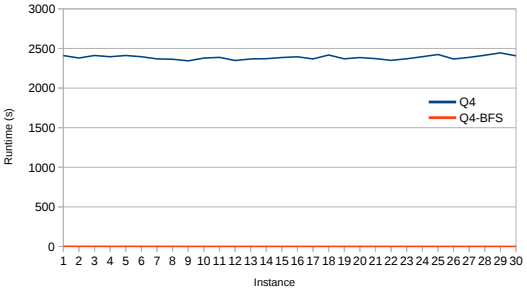



Figure 30: Results for query 4 QUERY4 (average runtime 2390.44 seconds) and the optimization approach 1 GRAPH-BFS (average runtime 1.65 seconds). The speedup factor is **1453**.

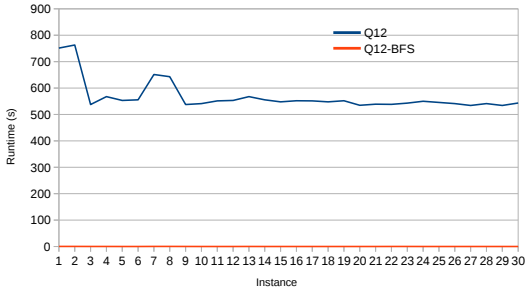



Figure 31: Results for query 12 QUERY12 (average runtime 567.44 seconds) and the optimization approach 1 GRAPH-BFS (average runtime 0.14 seconds). The speedup factor is **3838.77**.

# Optimization of Retrieval Algorithms on Knowledge Graphs

- (1) *Which author was the first to state that {Entity1} has an enhancing effect on {Entity2}?*

  (Q1) `match (n:Entity preferredLabel: "APP") -[r:hasRelation function: "increases"]-> (m:Entity preferredLabel: "gamma Secretase Complex"), (doc:Document documentID: r.context) <-[r2:isAuthor]- (author:Author) return doc, author order by doc.publicationDate limit 1'`

- Exclude the sorting functions

  (Q1-1) `match (n:Entity preferredLabel: "APP")-[r:hasRelation function: "increases"]->(m:Entity preferredLabel: "gamma Secretase Complex") return n,r,m`

- (15) *How many sources are there for the statements of a contradictory BEL statement?*

  (Q15) `match (e1:Entity) -[r1:hasRelation {function:"increases"}]-> (e2:Entity), (e1) -[r2:hasRelation {function:"decreases"}] -> (e2) return distinct e1.preferredLabel, e2.preferredLabel, count(r1) as ‘increases‘, count(r2) as ‘decreases‘ order by count(r1) desc`

- Exclude the sorting functions

  (Q15-1) `match (e1:Entity) -[r1:hasRelation function:"increases"]-> (e2:Entity), (e1) -[r2:hasRelation function:"decreases"]-> (e2) return distinct e1.preferredLabel, e2.preferredLabel`

# Optimization of Retrieval Algorithms on Knowledge Graphs

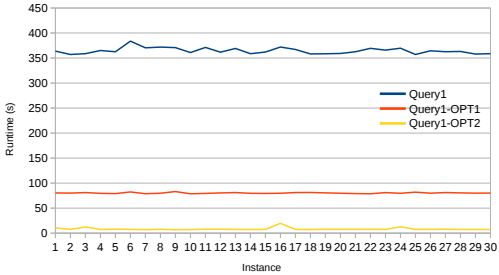



Figure 32: Results for query 1 Query1 (average runtime 364.45 seconds) and the optimization approaches 1 Query1-opt1 (average runtime 80.2 seconds) and 2 Query1-opt2 (average runtime 9.6 seconds). In total the speedup factor is **43.8**.

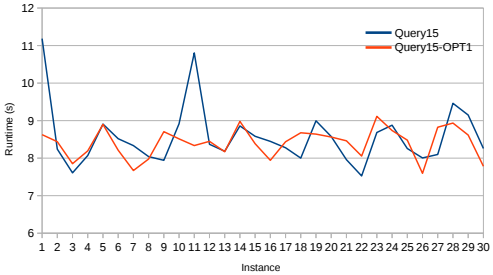



Figure 33: Results for query 15 Query15 (average runtime 8.6 seconds) and the optimization approach 1 Query15-opt1 (average runtime 8.4 seconds).

# Optimization of Retrieval Algorithms on Knowledge Graphs

- The most striking results are obtained with more complex queries.
- Although **indexes** for search performance can be used, they can not cover every possible combination for any query.
- Data integration needs to be done explicit. The more players, the harder the game will be.
- Collecting real world use cases and their complexity and runtime.
- Not all questions are graph-related.

Summary

- Knowledge Graphs play an important role in different application domains.
- Although a graph-theoretic definition exists and LPGs provide a feasible framework for graph databases, there is often much confusion.
- Analysis and structures of KGs
- Network Dynamics
- Graph Theory and AI
- Technical challenges