Graph Data Analysis & Exploration

— Graph Structure Analysis —

Matteo Lissandrini – Aalborg University

AALBORG UNIVERSITY

Outline

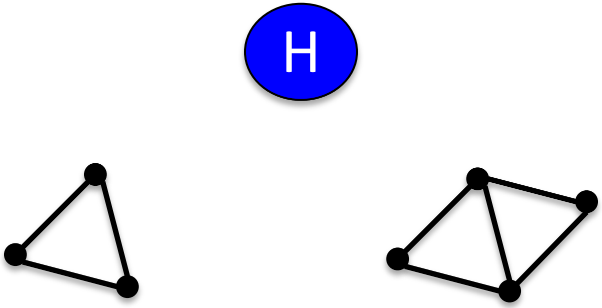
1. **Graph Structure Mining**
   - Graph homomorphism/isomorphism
   - Simulation/Strong Simulation
   - Graphlets/Motifs
   - Triangle Counting
2. **Frequent Graph Pattern Mining**
   - Graph-level subgraph frequency
   - Frequent Subgraph Mining
   - Extracting Shapes

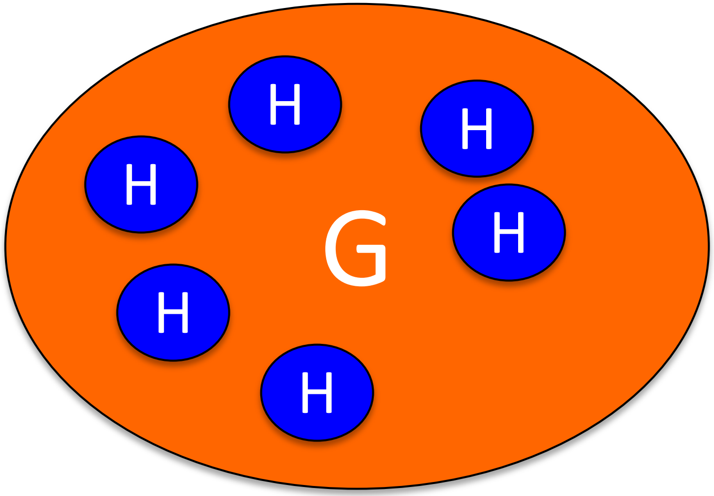
3. **Graph Clustering & Communities**
   - Connected components & Cliques
   - K-core & K-core decomposition
   - Community detection
   - Graph Partitioning & Conductance
   - Girvan Newman algorithm
   - Overlapping Communities
4. **Network Sampling**
   - Scale-down / Back-in-time sampling
   - Induced vs. Incident graph sampling
   - Biased Node Sampling
   - Shape Sampling

Graph Structure Mining

— Subgraph and Motif Search

Graph Structure Matching



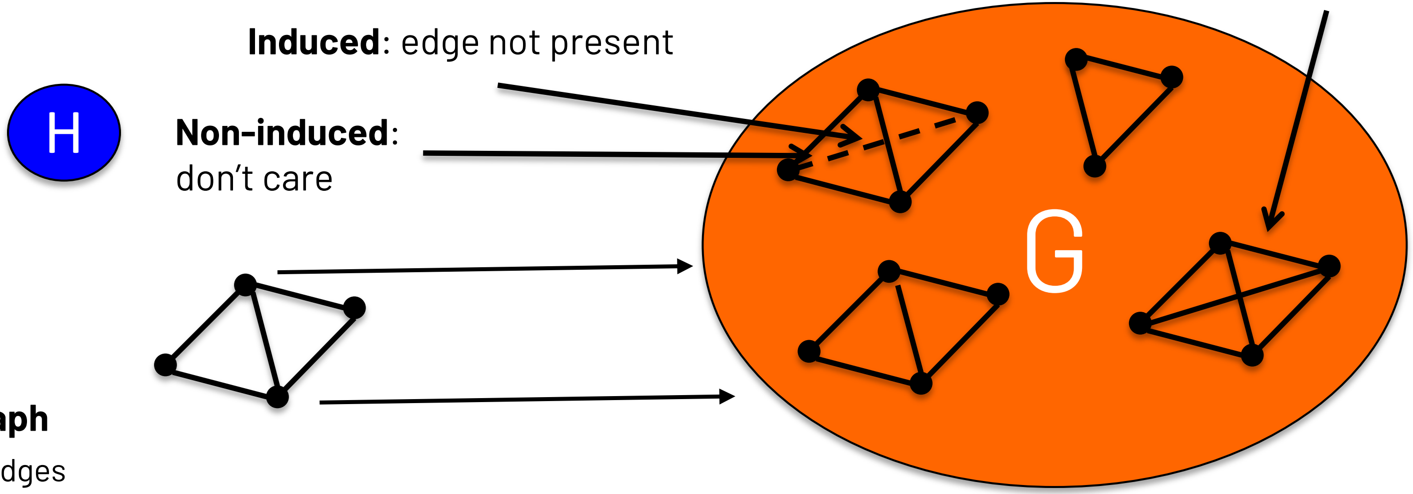


- G is a large graph (the input)
- H is a small "pattern" graph (the query)
- Count/find all occurrences of H in G
- Other names: graphlet analysis, motif counting

Thanks to C. Seshadhri
https://users.soe.ucsc.edu/~sesh/

What is a match?

Any set of 5 edges gives non-induced H



Induced H-subgraph

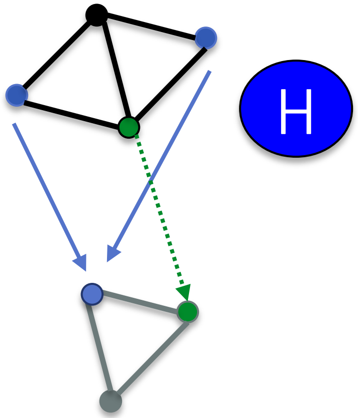
- Must match non-edges

Non-induced H-subgraph

- Don't care about non-edges

Subgraph-isomorphism vs Homomorphism

- 1-1 mapping of vertices vs many-1



Thanks to C. Seshadhri
https://users.soe.ucsc.edu/~sesh/

Homomorphism vs. Isomorphism

An **isomorphism** between two graphs G and H is a <u>bijective mapping</u>

$$\phi : G \mapsto H$$

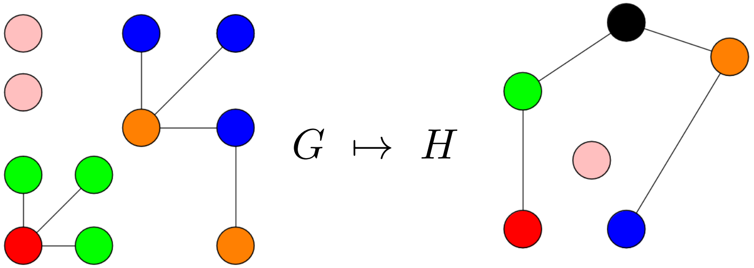
such that

$$(x, y) \in E(G) \Leftrightarrow (\phi(x), \phi(y)) \in E(H)$$

A **homomorphism** between two graphs G and H is a mapping

$$(x, y) \in E(G) \Rightarrow (\phi(x), \phi(y)) \in E(H)$$

Isomorphism is a stricter condition, while homomorphism just preserves edges in one direction



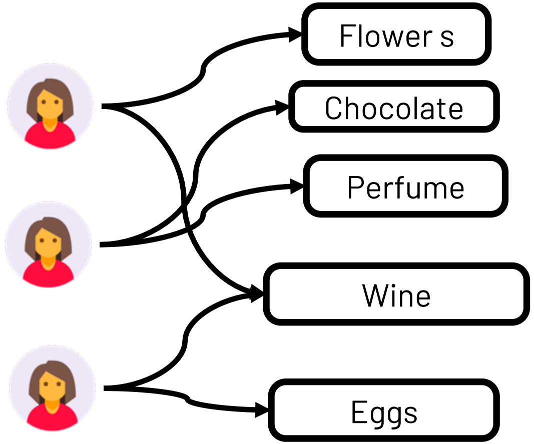
http://buzzard.ups.edu/courses/2013spring/projects/davis-homomorphism-ups-434-2013.pdf

Homomorphism vs. Isomorphism (2)

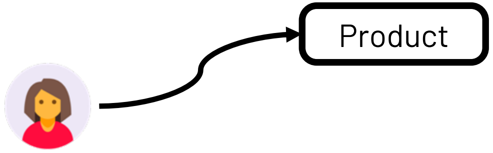
$$\phi \; : G \mapsto H$$

$$(x, y) \in E(G) \Rightarrow (\phi(x), \phi(y)) \in E(H)$$

G: Bi-partite graph



H: Schema of bi-partite graph



A bipartite graph is homomorphic to
the 2-nodes 1-edge graph that describes its schema.

The schema is ***<u>a form of summary of the graph</u>***

Isomorphism vs. Subgraph-isomorphism

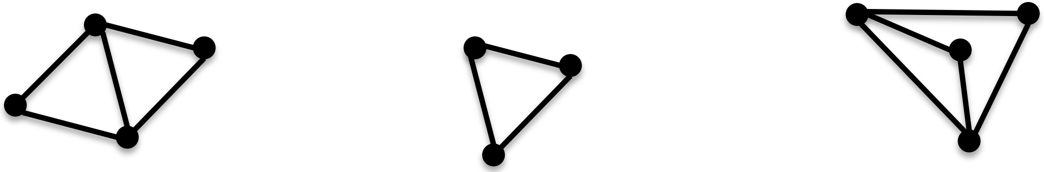
As decision problem:

- **Isomorphism** between G and H:
  is G *isomorphic to* H *?*

Subgraph-Isomorphism is NP-complete

Isomorphism is ¯_(ツ)_/¯

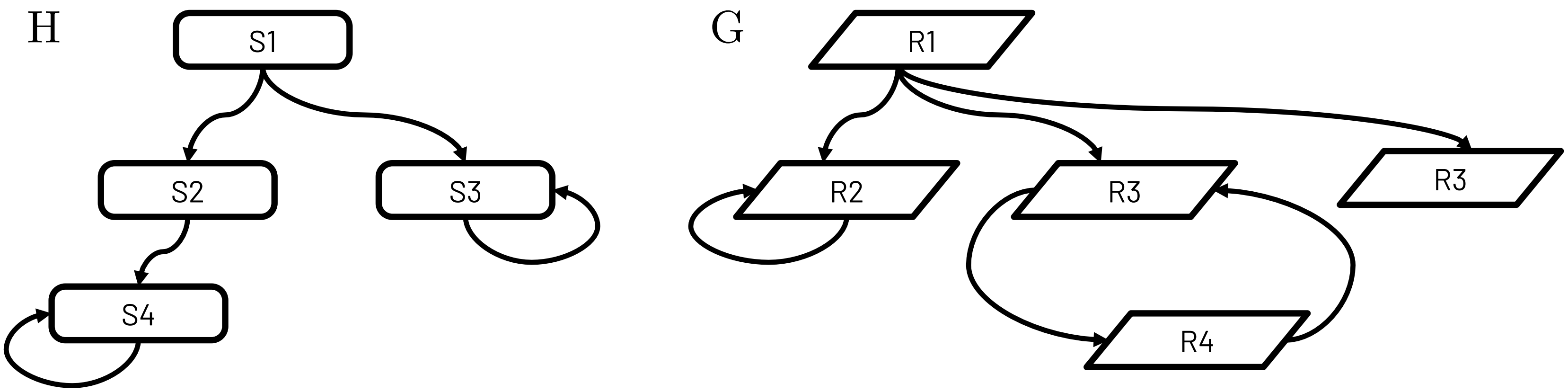
- **Subgraph-isomorphism** between G and H:
  is there a subgraph $H_0 \sqsubseteq H$ *such that* G *is isomorphic to* H_0



Complexity classes are tricky:

In reality, the subgraph isomorphism problem can be considered to be solvable in polynomial time in size of target graph if we consider that in practice the query graph has a bounded size!

Graph Simulation Matching



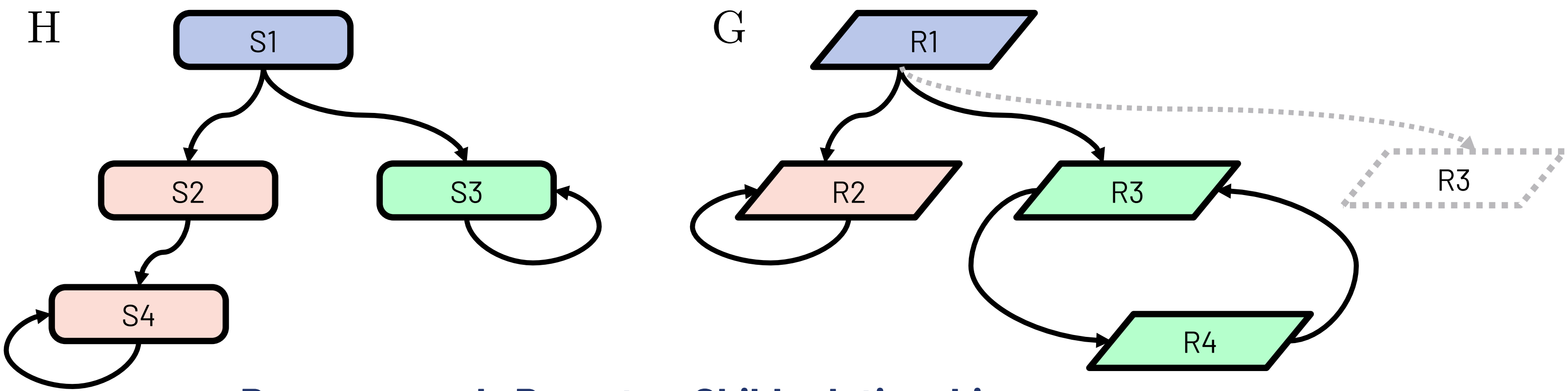
A graph G can simulate a graph H if there exists **a binary relation** $Sim \subseteq V_G \times V_H$ where for each $(u,v) \in Sim$ *if* $(v, v') \in E_H$ there is (u,u') in E_G such that $(u',v') \in Sim$

If I can follow one transition (edge) in one graph, the other should also be able to follow in the same way

https://dl.acm.org/doi/pdf/10.1145/2528937

Graph Simulation Matching

Simulation is computable in Polynomial time



Preserves only Parent → Child relationships

A graph G can simulate a graph H if there exists **a binary relation** $Sim \subseteq V_G \times V_H$ where for each $(u,v) \in \mathit{Sim}$ *if* $(v, v') \in E_H$ there is (u,u') in E_G such that $(u',v') \in \mathit{Sim}$

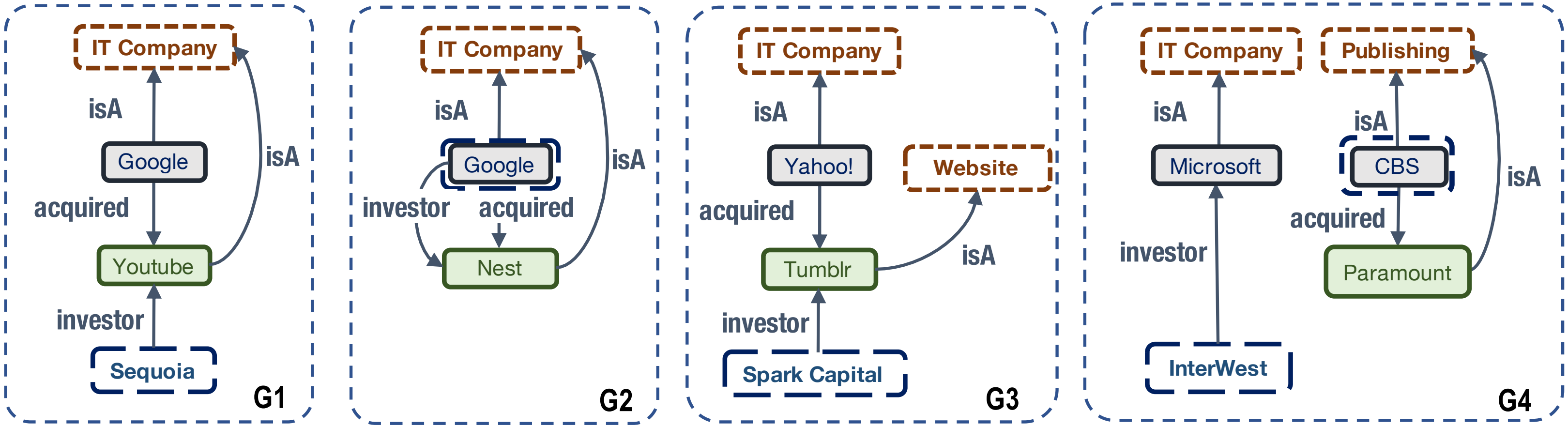
Usually, Simulation takes into consideration Node/Edge Labels which I've ignored in my example!

https://dl.acm.org/doi/pdf/10.1145/2528937

Graph Isomorphism vs. Simulation Variants

Structural Congruence/Similarity

Isomorphism requires an bijective **function**
Simulation requires only a **parent-child** edge preserving **relation**
Strong Simulation requires also **child-parent**, **connectivity** and **limited diameter**

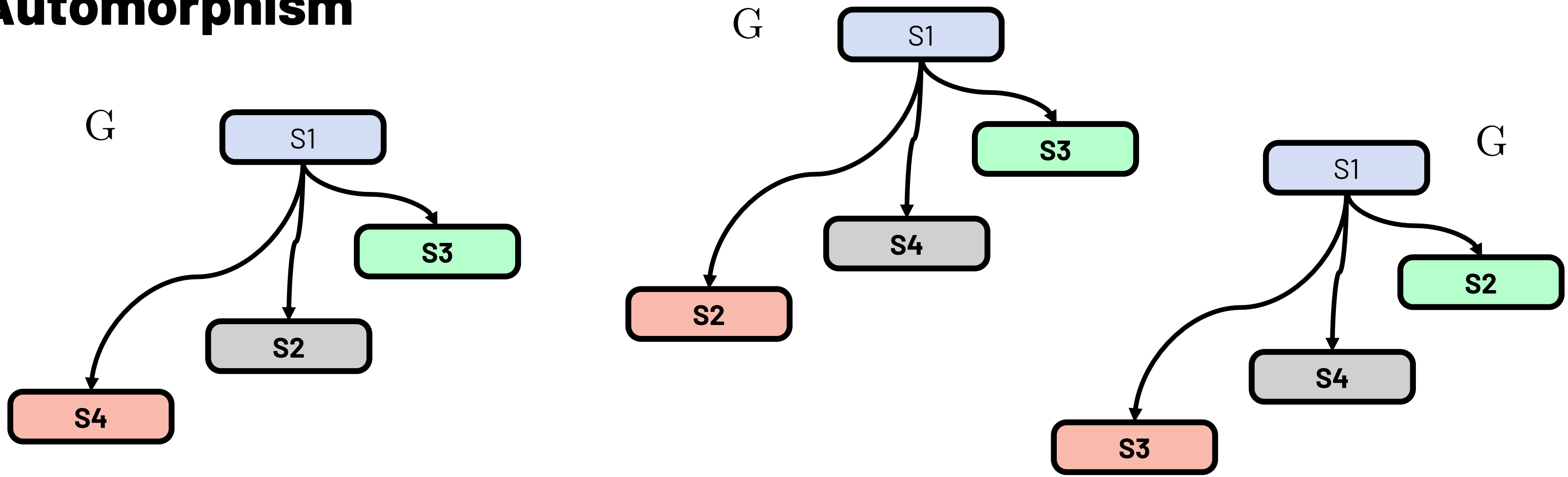


Example of Simulating (G1~ {G2,G3,G4}) and Strong-simulating Graphs (G1≈G2)

Strong Simulation preserves close connectivity

Strong simulation: Capturing topology in graph pattern matching
– Shuai Ma et al., 2014

Automorphism



- A graph G is trivially isomorphic to itself
  - so it exists always at least one bijective mapping $\phi : G \mapsto G$
  - but what if there exist more than one $\phi_1 \neq \phi_2 \neq \ldots : G \mapsto G$?
- Automorphism: an homomorphism from a graph to itself

Challenge: *A Graph Structure Matching Algorithm needs to account for automorphism, and possibly avoid duplicate computation*

Analyzing Network Motifs

- **Network Motif:** a recurrent, significant node-induced subgraph structure in a network (graph)
  - **Recurrent → Frequent:** How to identify "frequency"?
  - **Significant → appears more often than in a random graph**

Z_i is the **significance score** of S_i
negative value indicates
under-representation

$$Z_i = \frac{N_i^{\text{real}} - \bar{N}_i^{\text{rand}}}{\text{std}\left(N_i^{\text{rand}}\right)}$$

Average frequency of S_i
in random graphs

- **Motifs analysis is based on network structure** (i.e., not considering labels)

What are the **possible motifs** that can appear in
a graph given a fixed number of nodes K?
They are called **Graphlets**

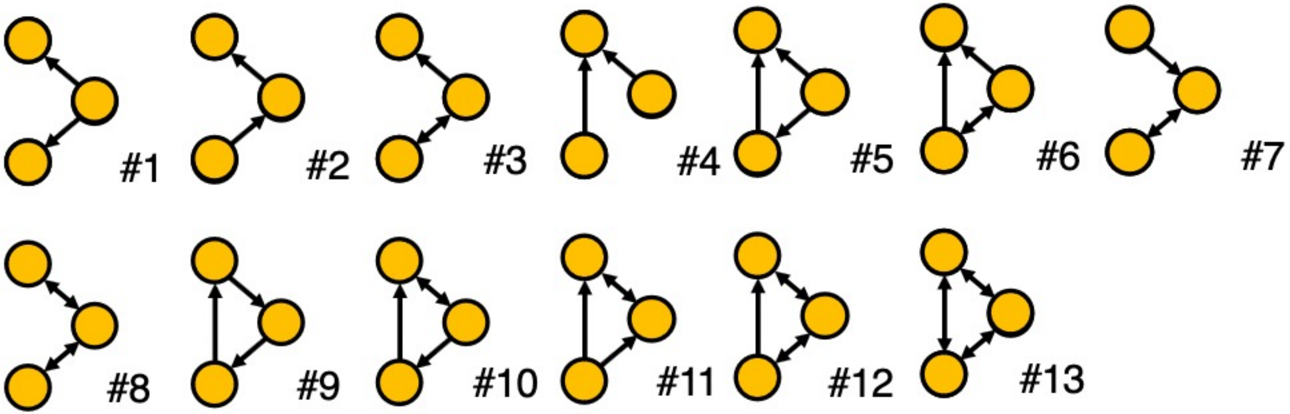
http://web.stanford.edu/class/cs224w/slides/12-motifs.pdf

Graphlets

A graphlet is class of connected isomorphic subgraphs of a given size.

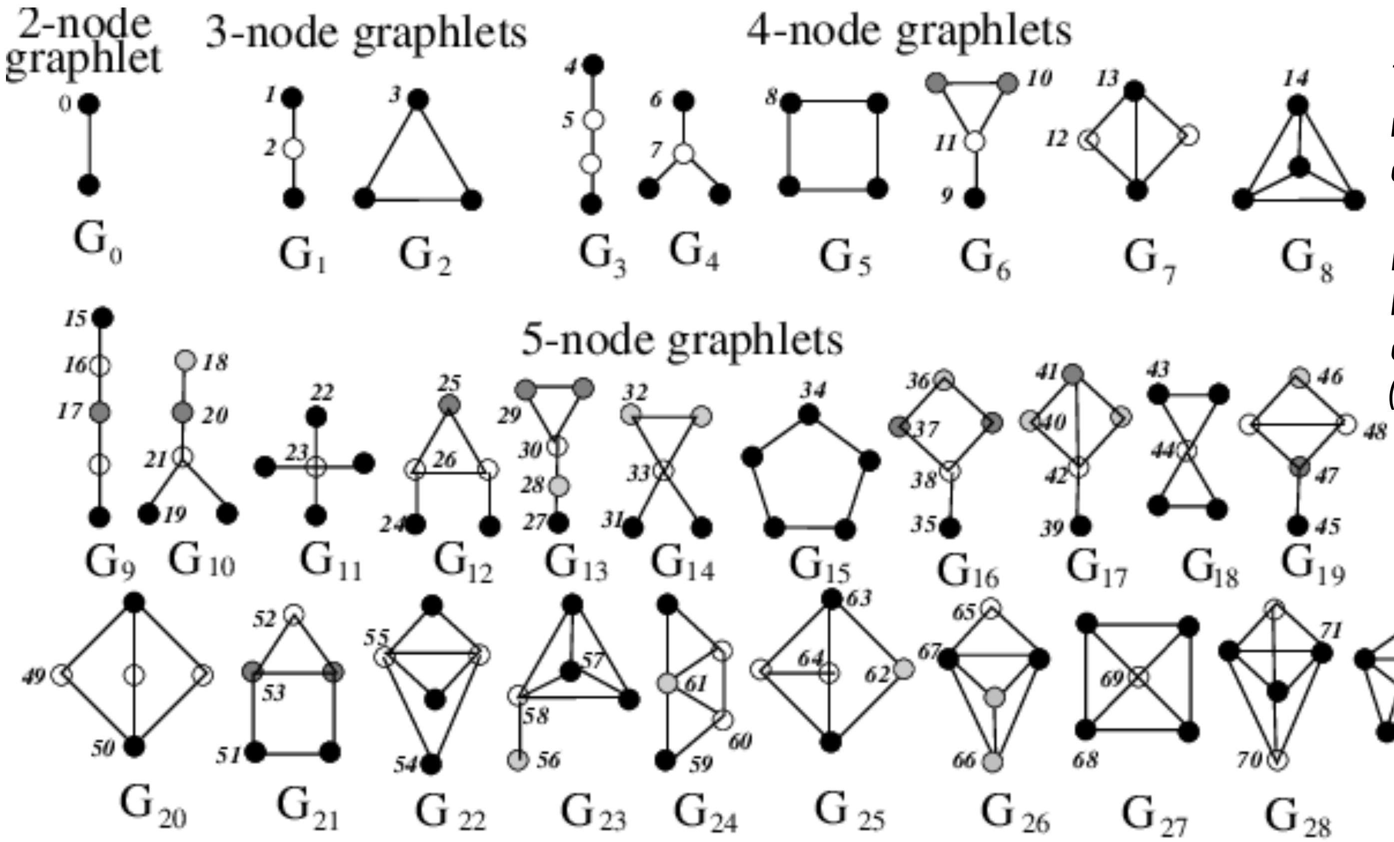
Two graphlet occurrences are isomorphic, whereas two occurrences of two distinct graphlets are non-isomorphic

Example: all non-isomorphic, connected, directed graphltes of size 3



http://web.stanford.edu/class/cs224w/slides/12-motifs.pdf

There are 30 Graphlets of size ⩽5 but ~11M of size 10



The thirty 2-, 3-, 4-, and 5-node graphlets and their automorphism orbits

In a graphlet G_i, nodes belonging to the same orbit are of the same shade (Pržulj, 2006).

Pržulj, Nataša. "Biological network comparison using graphlet degree distribution." *Bioinformatics* 23.2 (2006): e177-e183.

Outline

1. **Graph Structure Mining**
   - Graph homomorphism/isomorphism
   - Simulation/Strong Simulation
   - Graphlets/Motifs
   - Triangle Counting
2. **Frequent Graph Pattern Mining**
   - Graph-level subgraph frequency
   - Frequent Subgraph Mining
   - Extracting Shapes

3. **Graph Clustering & Communities**
   - Connected components & Cliques
   - K-core & K-core decomposition
   - Community detection
   - Graph Partitioning & Conductance
   - Girvan Newman algorithm
   - Overlapping Communities
4. **Network Sampling**
   - Scale-down / Back-in-time sampling
   - Induced vs. Incident graph sampling
   - Biased Node Sampling
   - Shape Sampling

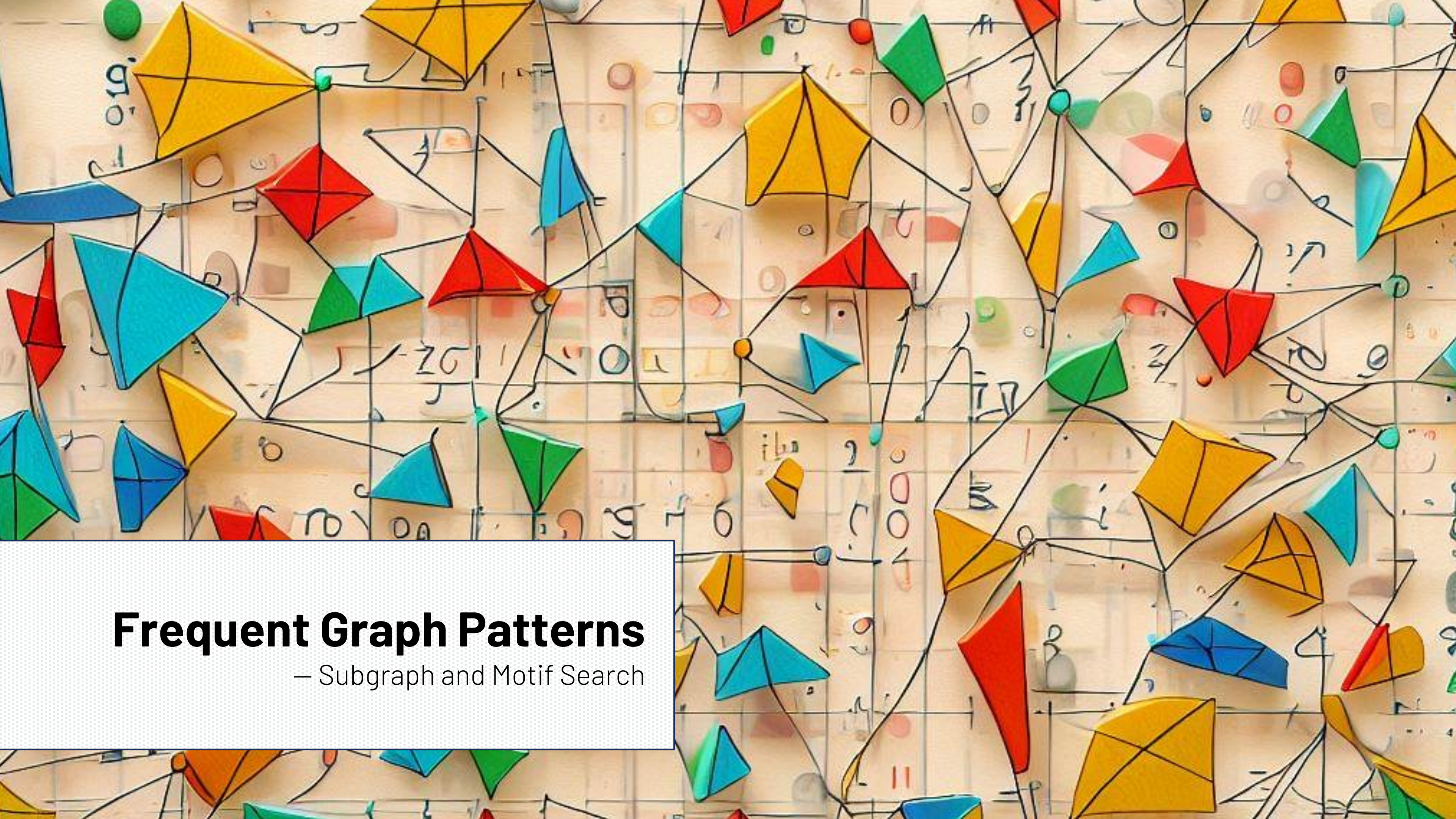
Outline

1. **Graph Structure Mining**
   - Graph homomorphism/isomorphism
   - Simulation/Strong Simulation
   - Graphlets/Motifs
   - Triangle Counting

2. **Frequent Graph Pattern Mining**
   - Graph-level subgraph frequency
   - Frequent Subgraph Mining
   - Extracting Shapes

3. **Graph Clustering & Communities**
   - Connected components & Cliques
   - K-core & K-core decomposition
   - Community detection
   - Graph Partitioning & Conductance
   - Girvan Newman algorithm
   - Overlapping Communities

4. **Network Sampling**
   - Scale-down / Back-in-time sampling
   - Induced vs. Incident graph sampling
   - Biased Node Sampling
   - Shape Sampling



Frequent Graph Patterns

— Subgraph and Motif Search

A detour: Association Rules

- **If someone buys diapers and milk, then they are likely to buy beer**
  - Don’t be surprised to find beers placed next to diapers!

Supermarket shelf management **– Market-basket model:**

- **Goal:** Identify items that are **bought together** by sufficiently many customers
- **Approach:** Process the sales data collected with barcode scanners to **find dependencies among items**

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets,
http://www.mmds.org

The Market-Basket Model

- A large set of **items**
  - e.g., things sold in a supermarket
- A **large set** of **baskets**
- Each basket is a **small subset of items**
  - e.g., the things one customer buys on one day
- Want to discover **association rules**
  - People who bought {x,y,z} tend to buy {v,w}

Input:

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Output:

Rules Discovered:
{Milk} --> {Coke}
{Diaper, Milk} --> {Beer}

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Frequent Itemsets

- **Simplest question:** Find sets of items that appear together "frequently" in baskets
- ***Support*** for itemset I: Number of baskets containing all items in I
  - (Often expressed as a fraction of the total number of baskets)
- Given a ***support threshold*** s,
  the sets of items that appear in at least s baskets
  are called ***frequent itemsets***

A-priori principle:

if {A,B,C} is **frequent** → {A,B} is also **frequent**

if {A,B} is **not frequent** → {A,B,C} **cannot** be frequent

Input:

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Support of
{Beer, Bread} = 2/5 = 0.4

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Frequent Graph Pattern Mining

- Frequent subgraphs
  - **A (sub)graph is *frequent* if its support (occurrence frequency) in a dataset is no less than a minimum support threshold**

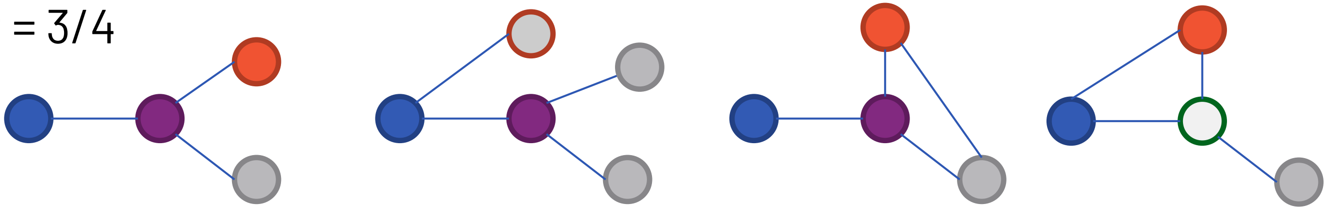
- Applications of graph pattern mining:
  - Mining biochemical structures
  - Program control flow analysis
  - Mining Social/Web communities
  - Building blocks for graph classification, clustering, compression, comparison

Graph Pattern Mining - Database (Set) of graphs

Problem

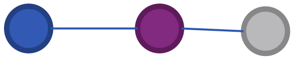
Given a set of graphs $\{G_1, G_2, \ldots, G_N\}$ find pattern P that appear at least in σ of them

Min support = 3/4

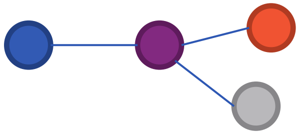


G1 G2 G3 G4

Frequent

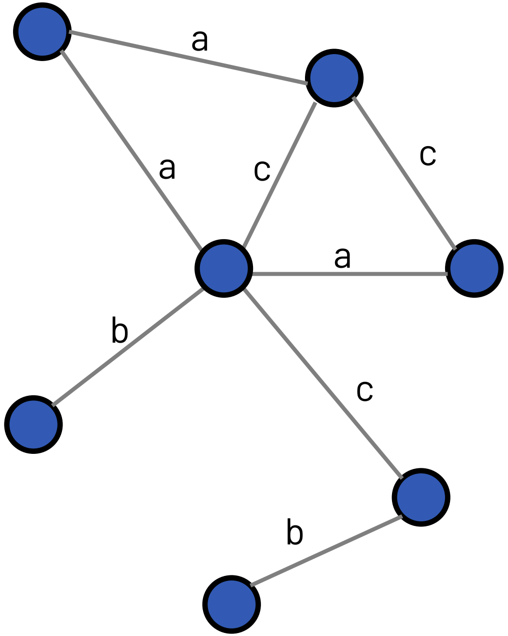


Non-Frequent



Support: frequency of a subgraph appearing in a **set** of graphs

Frequent Subgraph Mining – Single Large Graph



G

Problem
Find all subgraphs of G that appear at least σ times

Suppose $\sigma = 2$, the frequent subgraphs are (only edge labels)

- a, b, c
- a-a, a-c, b-c, c-c
- a-c-a ...

Exponential number of patterns!!!

Support over a single large graph

A support measure is **admissible** if for any pattern P and any sub-pattern $Q \sqsubset P$ the **support $\boldsymbol{\sigma(P)}$ is not larger** than support of Q, i.e., $\sigma(Q) \geq \sigma(P)$.
This is also referred **as anti-monotonicity** property.

Problem: The number of occurrences of a pattern in a single large graph is not an admissible support measure

$P1 \sqsubset P2$, but $1 = \sigma(P1) < \sigma(P2) = 2$

How can we solve this problem?

Support over a single large graph

A number of admissible support measures have been proposed:

1. **Maximum Independent Set support** (MIS)
   - Based on maximum number of **non-overlapping matches**
2. **Harmful overlap support** (HO)
   - Based on the number of patterns for **which no (multi-node) subgraph** is identical
3. **Minimum Image-based support** (MNI)
   - Based on the **number of times a node in the pattern** is mapped into a distinct node in the graph

Maximum independent set support (MIS)

- **Idea:** Count how many times a pattern is mapped to a **non-overlapping subgraph**

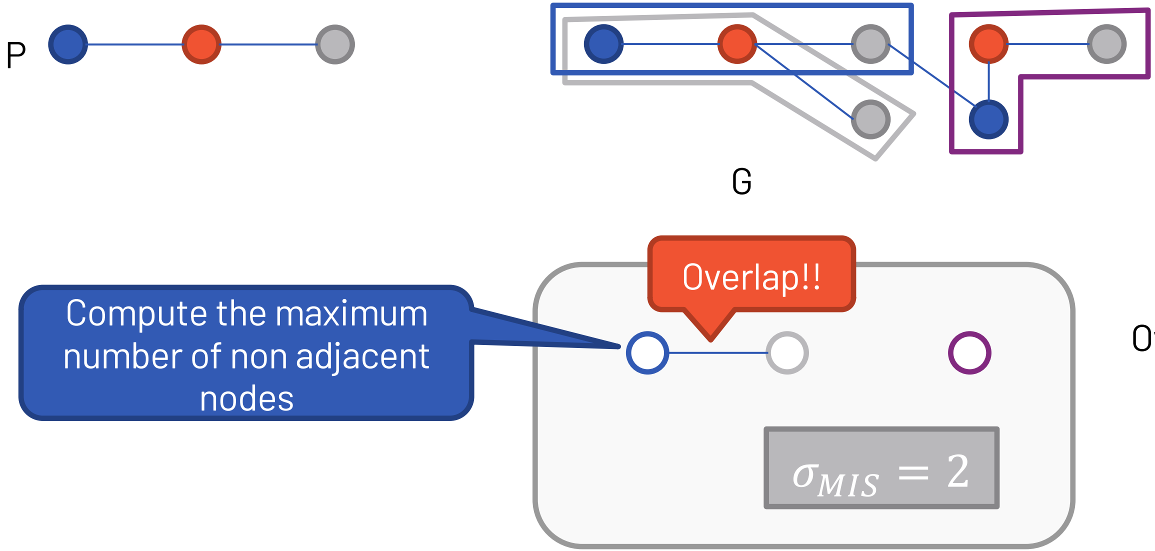
MIS computation

1. Construct an **overlap graph** $\mathrm{G_O}: \langle \mathrm{V_O}, \mathrm{E_O} \rangle$, in which the set of nodes $\mathrm{V_O}$ is the set of matches of a pattern P into a graph G and $E_O = \{(f_1, f_2) | f_1, f_2 \in V_O \wedge f_1 \neq f_2 \wedge f_1 \sqcap f_2 \neq \emptyset\}$, i.e., $\mathrm{E_O}$ has an an edge among each pair of overlapping matches.

2. σ_{MIS} = size of the **maximum independent set** of nodes in the overlap graph.

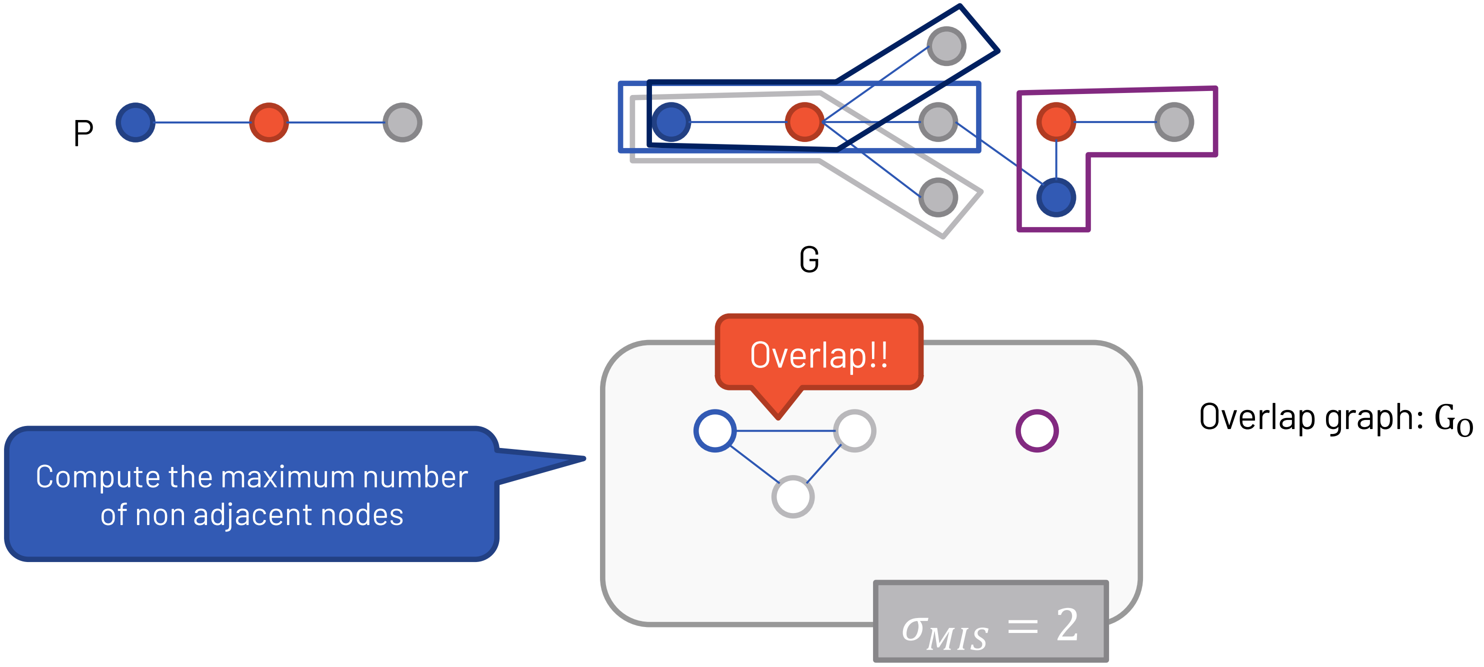
- An **independent set** of nodes in a graph is a subset of non-adjacent nodes of the graph, i.e. $\mathrm{G}: \langle V, E \rangle, I \subseteq V$ s.t. $\forall (u, v) \in I, (u, v) \notin E$
- The biggest possible independent set is called the **maximum independent set**

Maximum independent set support (MIS) : Example



Overlap graph: G_O

Maximum independent set support (MIS) : Example 2

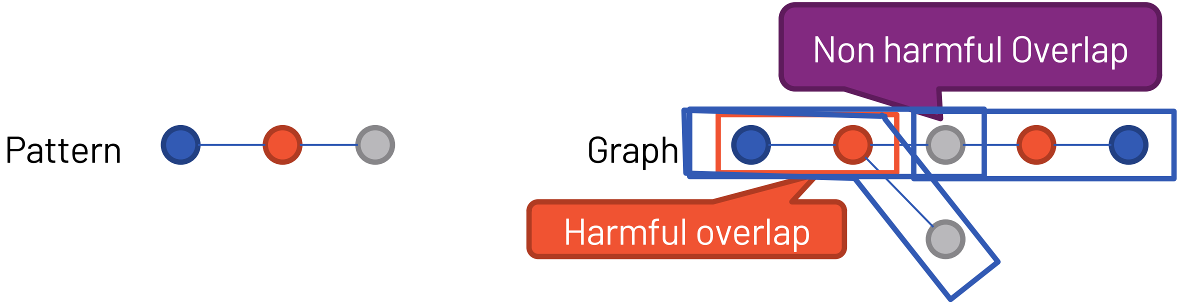


Finding a maximum independent set is NP-hard → Very High computational cost

Harmful overlap support (HO)

- MIS support can be very restrictive and considers overlap among *single nodes*

- Harmful overlap support σ_{HO} considers as *harmful* those *subgraphs* that share a ***common (multi-node) subgraph***



Support is computed like in MIS, the difference is how we compute the overlap graph

Fiedler, M. and Borgelt, C., 2007. Subgraph support in a single large graph. *ICDMW 2007.*

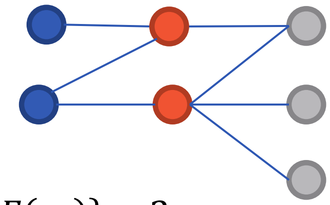
Minimum Image-based support (MNI) : Example

Simpler support based on the minimum number of times a node of the pattern is mapped to the graph

Let $f_1, \dots, f_m$ be the set of isomorphisms of a pattern $P: \langle V_P, E_P, \ell_P \rangle$ in a graph G. Let $F(v) = |\{f_1(v), \dots, f_m(v)\}|$ be the number of distinct mappings of a node $v \in V_P$ to a node in G by functions $f_1, \dots, f_m$. The **Minimum Image-based support (MNI)** $\sigma_{MNI}(P)$ of P in G is $\sigma_{MNI}(P) = \min\{F(v), v \in V_P\}$

Pattern

Graph



$F(v_1) = 2$

$F(v_2) = 2$

$F(v_3) = 3$

$$\sigma_{MNI}(P) = \min\{F(v_1), F(v_2), F(v_3)\} = 2$$

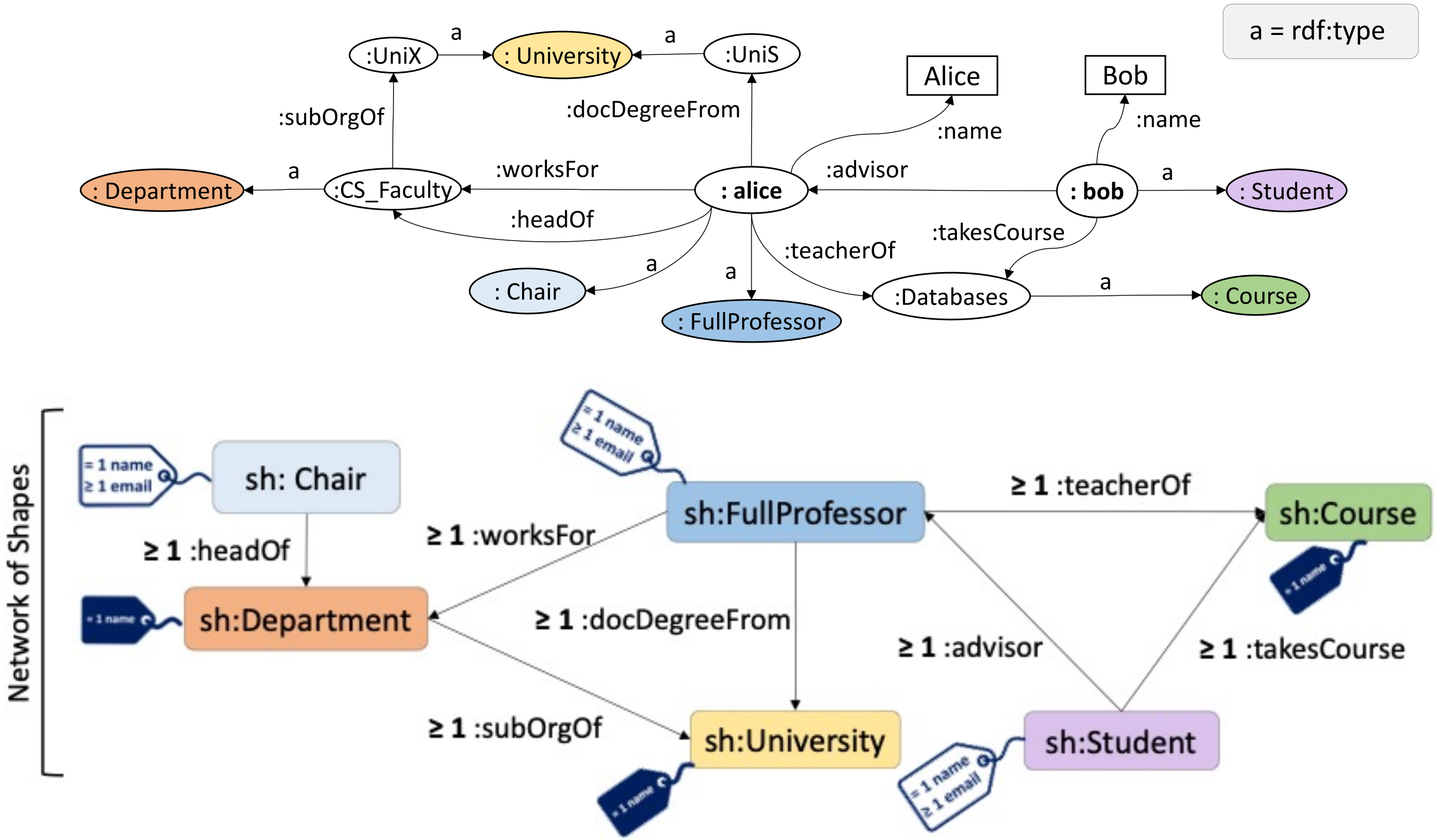
Chen, C., Yan, X., Zhu, F. and Han, J. gapprox: Mining frequent approximate patterns from a massive network. ICDM, 2007.

Properties of the support measures

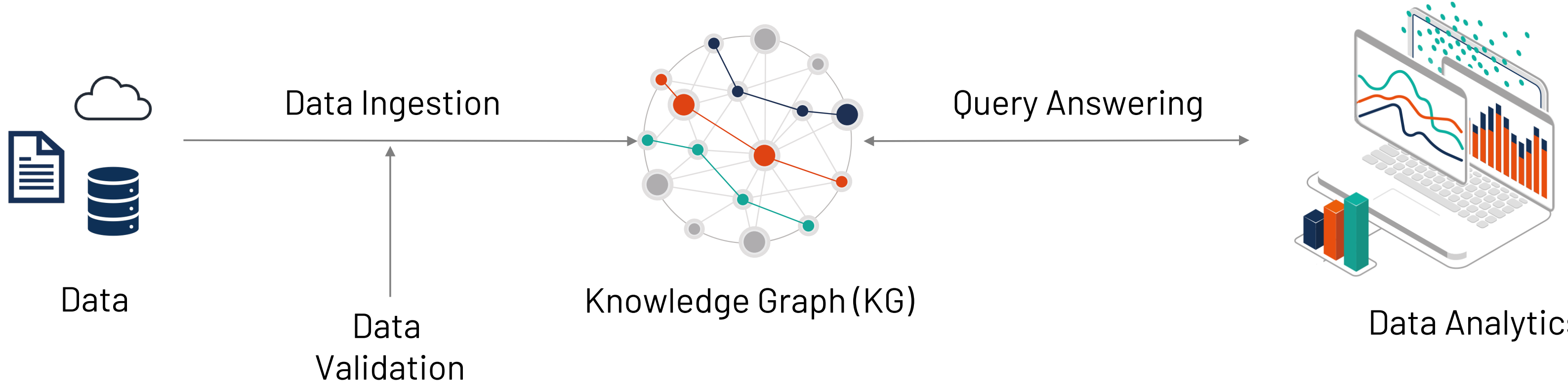
- MIS, HO, and MNI are anti-monotone (proof omitted)
- Computation time:
  - MIS and HO are **NP-**hard to compute (*there is no algorithm to solve them efficiently*)
  - MNI can be computed in polynomial time!
- What is the relationship among the support sets?
  - MIS support set is a subset of HO support set, which is a subset of MNI support set
  - $\sigma_{MIS}(P) \leq \sigma_{HO}(P) \leq \sigma_{MNI}(P)$

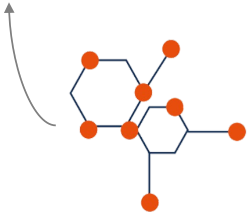
Mohammed Elseidy, Ehab Abdelhamid, Spiros Skiadopoulos, and Panos Kalnis. 2014. GraMi: frequent subgraph and pattern mining in a single large graph. Proc. VLDB Endow. 7, 7 (March 2014), 517–528.
https://github.com/ehab-abdelhamid/GraMi

Mining Graph Shapes: a.k.a. structural summarization



Validating Graph Data with Shapes





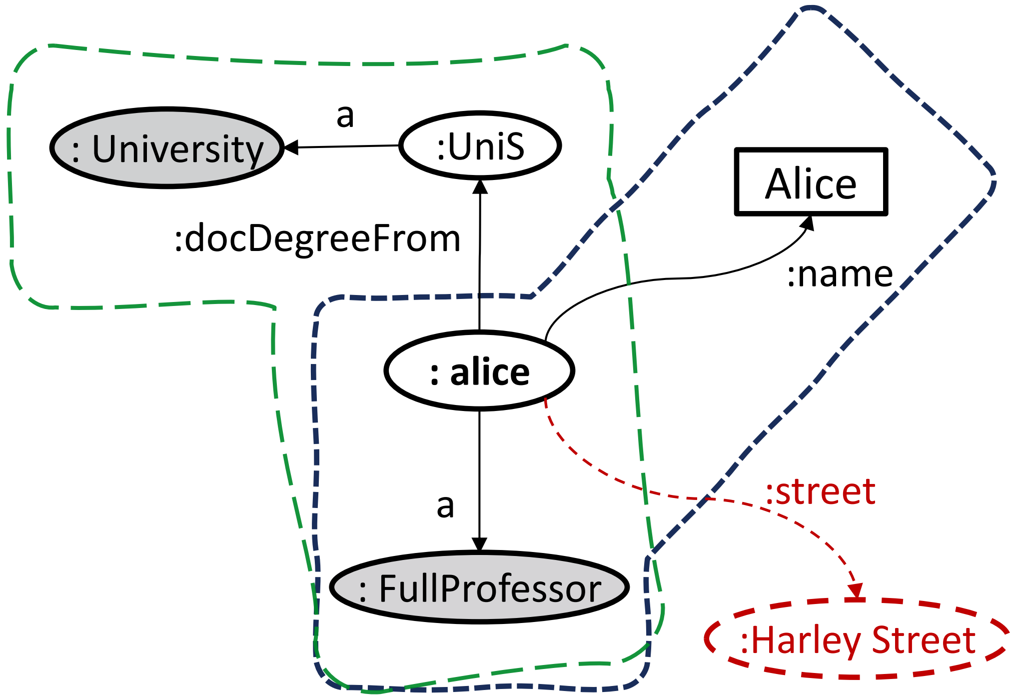
Validating Shapes

Shapes constraint language (**SHACL** – **W3C** standard)
Shapes expressions (**ShEx**)

A FullProfessor must have :

- *a name*
- *a degree from a University*

Mining Graph Shapes



Rabbani, Kashif, Matteo Lissandrini, and Katja Hose. "Extraction of Validating Shapes from very large Knowledge Graphs"- VLDB'23
https://relweb.cs.aau.dk/qse/

A FullProfessor must have :

- *a name*
- *a degree from a University*

Using Support and confidence allows to avoid extracting spurious shapes

Support

Support of **property shape** is defined as the **number** of entities conforming to its constraints.

Confidence

The **ratio** between ***how many entities conform to a specific constraint*** and the ***total number of entities for the target class of the node shape***.

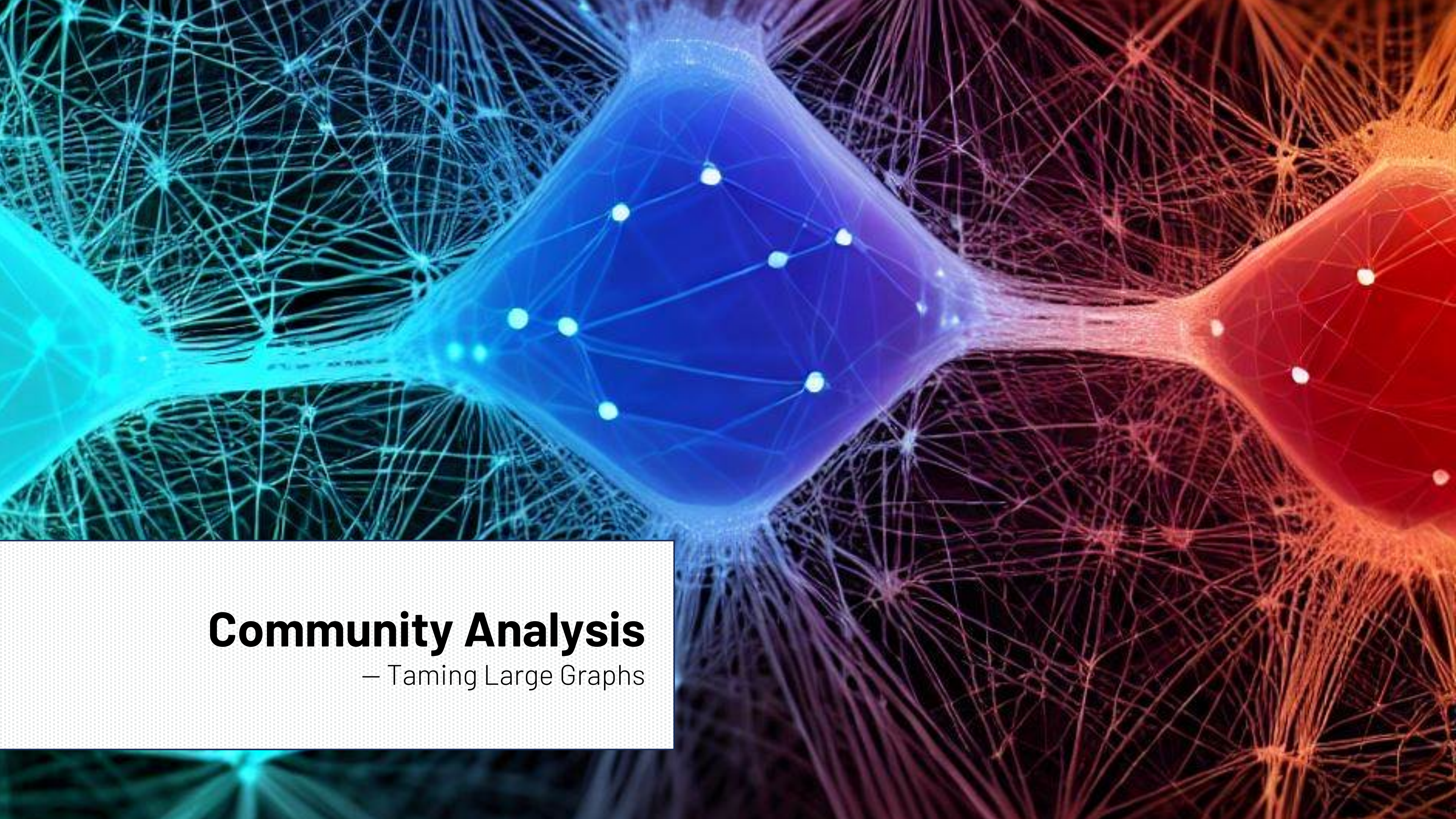
Outline

1. **Graph Structure Mining**
   - Graph homomorphism/isomorphism
   - Simulation/Strong Simulation
   - Graphlets/Motifs
   - Triangle Counting
2. **Frequent Graph Patterns**
   - Graph-level subgraph frequency
   - Frequent Subgraph Mining
   - Extracting Shapes

3. **Graph Clustering & Communities**
   - Connected components & Cliques
   - K-core & K-core decomposition
   - Community detection
   - Graph Partitioning & Conductance
   - Girvan Newman algorithm
   - Overlapping Communities
4. **Network Sampling**
   - Scale-down / Back-in-time sampling
   - Induced vs. Incident graph sampling
   - Biased Node Sampling
   - Shape Sampling

Outline

1. **Graph Structure Mining**
   - Graph homomorphism/isomorphism
   - Simulation/Strong Simulation
   - Graphlets/Motifs
   - Triangle Counting
2. **Frequent Graph Patterns**
   - Graph-level subgraph frequency
   - Frequent Subgraph Mining
   - Extracting Shapes
3. **Graph Clustering & Communities**
   - Connected components & Cliques
   - K-core & K-core decomposition
   - Community detection
   - Graph Partitioning & Conductance
   - Girvan Newman algorithm
   - Overlapping Communities
4. **Network Sampling**
   - Scale-down / Back-in-time sampling
   - Induced vs. Incident graph sampling
   - Biased Node Sampling
   - Shape Sampling



Community Analysis

— Taming Large Graphs

Reachability: Connected components

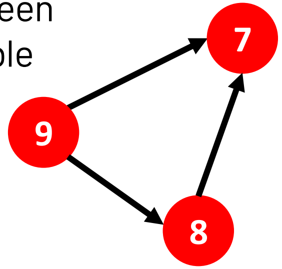
- **A connected component** is a portion of the graph where each node can reach all other nodes: pairwise reachable.

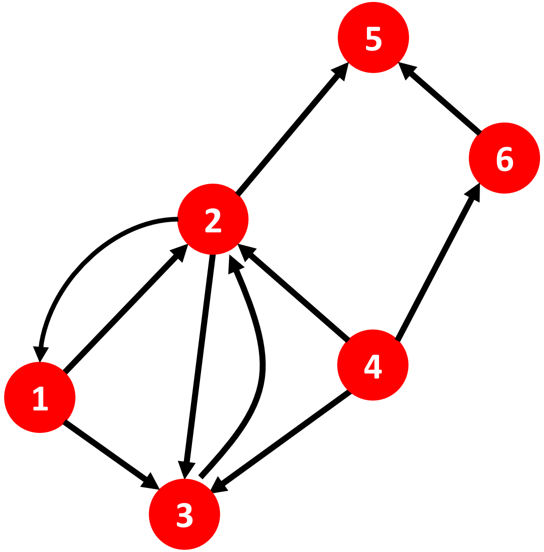
In a directed graph we can have connected components, but if we follow directions, then it may happen that we cannot reach all nodes.

- **A strongly connected component** is a portion of a directed graph where there is a directed path between any two nodes. All nodes are pairwise reachable when following directions.

- **A weakly connected component** is a portion of a directed graph where there is an **undirected** path between any two nodes. All nodes are pairwise reachable when **ignoring** directions.

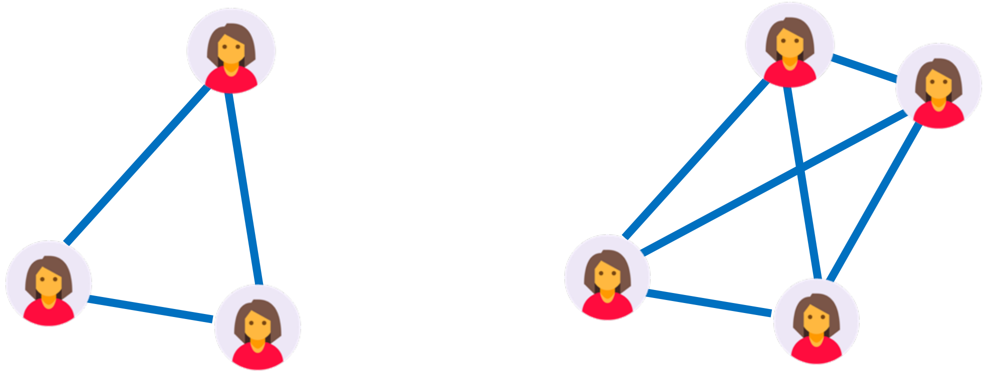
USE BFS :
1) Start from a node;
2) Obtain all reachable nodes and mark them;
3) Increment CC counter
4) Take next node not already marked, and start again



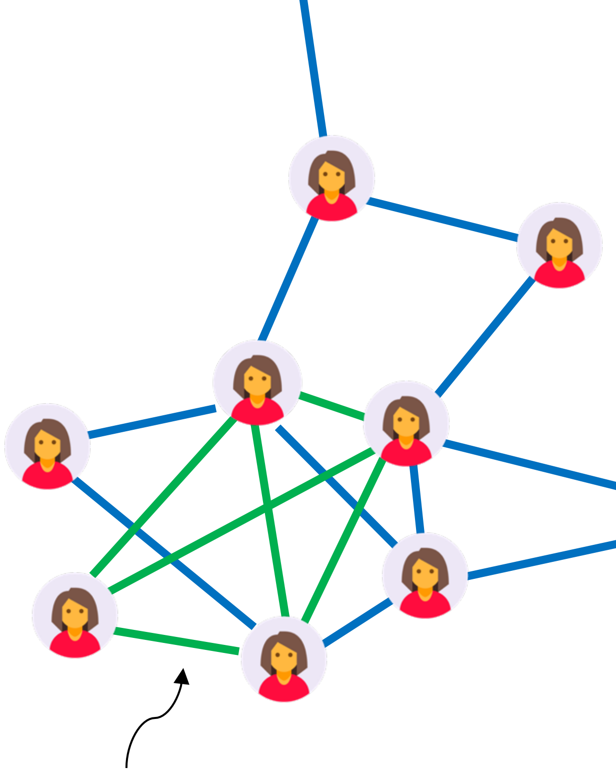


Cliques

Subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent



A clique has density 1



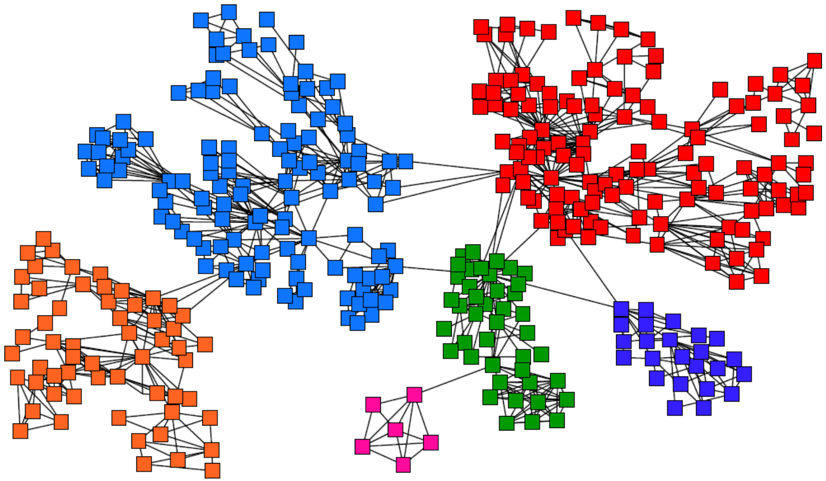
Complete subgraphs:
A portion of the graph forming a clique, each node is connected to each other node

Communities: locally dense connected subgraphs

Connected & Density Hypothesis: Communities are locally dense connected subgraphs in a network.

Connectedness Hypothesis:
Each community corresponds to a connected subgraph: a community cannot consist of two subgraphs that do not have a link to each other.

Density Hypothesis:
Nodes in a community are more likely to be connected to members of the same community than to nodes in other communities.



[BA01, CHAPTER] BARABÁSI, ALBERT-LÁSZLÓ, NETWORK SCIENCE, CHAPTER 9: COMMUNITIES
http://networksciencebook.com/chapter/9

Graph Partitioning

Graph Cut: *The number of edges that go from G_1 to G_2*

$$\mathbb{C} = |Cut(G_1, G_2)|$$

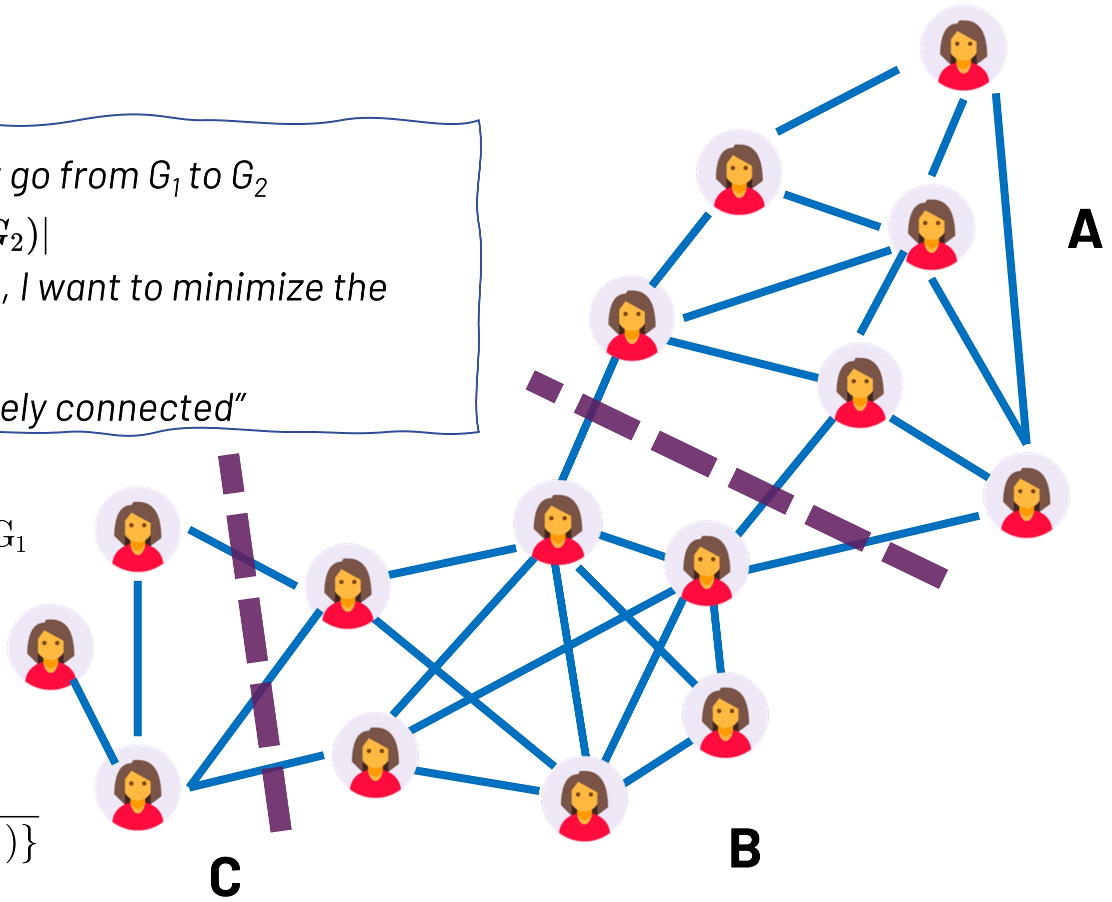
If we need to split the graph in 2 parts, I want to minimize the number of edges I "break".

But I also want the 2 parts to be "densely connected"

Conductance: given two portions of a graph G_1 and G_2, the conductance is the sum of the edges going from G_1 to G_2 divided the the minimum between the edges in G_1 and in G_2

We want to minimize the following

$$C\left(G_1, G_2\right) = \frac{\mathbb{C}}{\min\left\{\left(\mathbb{C} + |G_1|\right), \left(\mathbb{C} + |G_2|\right)\right\}}$$



Graph Partitioning (Example)

$$\mathbb{C} = |Cut(G_1, G_2)|$$

We want to minimize the following

$$C\left(G_1, G_2\right) = \frac{\mathbb{C}}{\min\left\{\left(\mathbb{C} + |G_1|\right), \left(\mathbb{C} + |G_2|\right)\right\}}$$

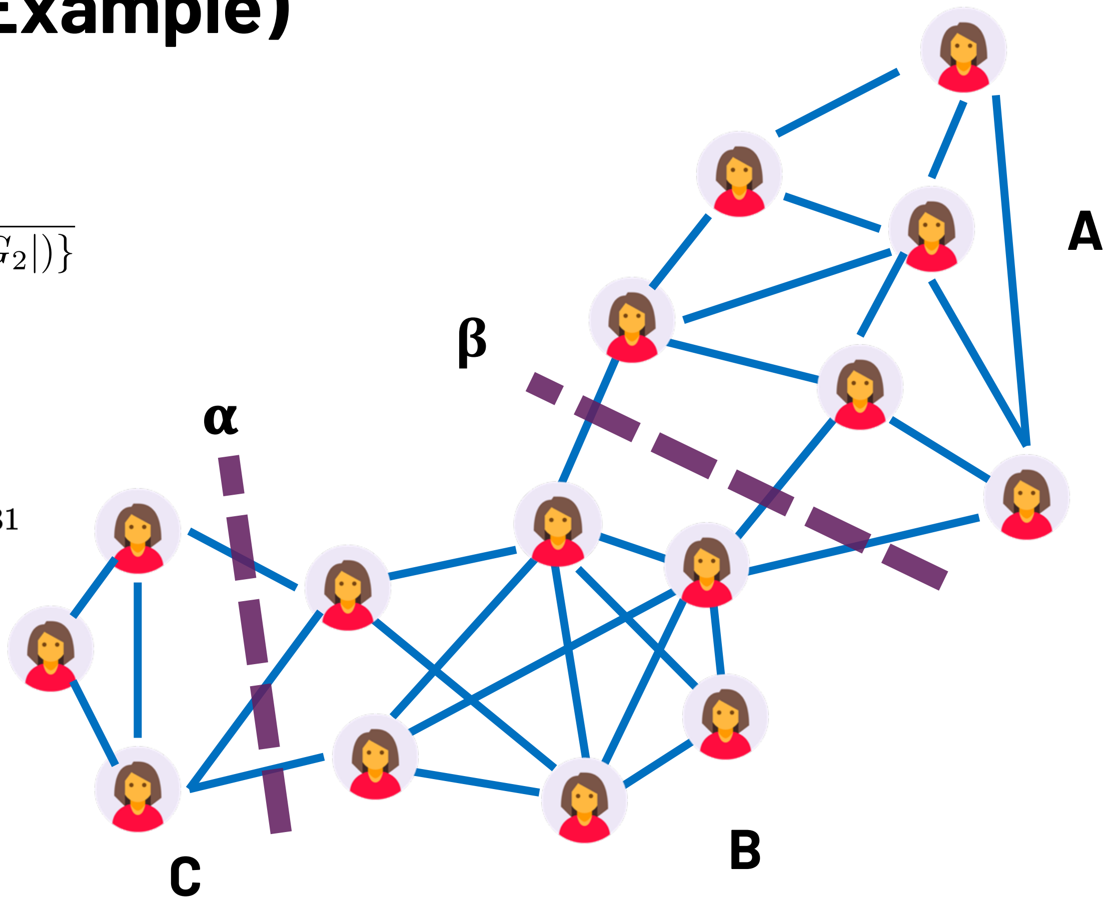
$$|\mathbf{A}| = \mathbf{10} \quad |\mathbf{B}| = \mathbf{10} \quad |\mathbf{C}| = \mathbf{3}$$

$$|\alpha| = \mathbf{3} \quad |\beta| = \mathbf{3}$$

$$\mathbf{C}(A, B + C) = \frac{3}{\min\{(3 + 10), (3 + 16)\}} \approx 0.231$$

$$\mathbf{C}(A + B, C) = \frac{3}{\min\{(3 + 23), (3 + 3)\}} \approx 0.5$$

Where should we cut?



Basic Centrality measures

How important is a node in a graph?

1. **Degree centrality:** number of neighbors of node v

2. **Closeness centrality:** reciprocal of the total distance from a node v to all the other nodes in a network

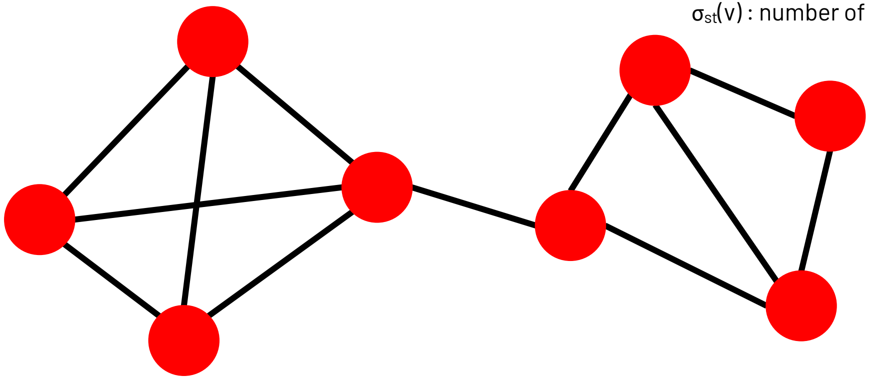
3. **Betweenness centrality:** ratio of the number of shortest paths passing through a node v out of all shortest paths between all node pairs in a network

$$C_c(v) = \frac{1}{\sum_{u \in V} \delta(u,v)}$$

$\delta(u,v)$ is the distance between node u and v.

$$C_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

σ_{st} : number of shortest paths between node s and t
$\sigma_{st}(v)$: number of shortest paths passing on a node v out σ_{st}



Connected graphs: These measure have meaning only when referring to a connected graphs

Community Detection

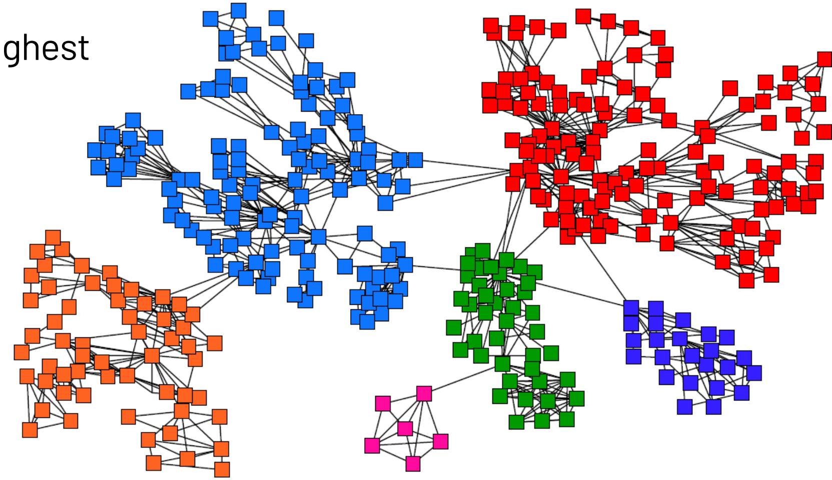
Community: portion of a graph with high internal connectivity *(also called modules or clusters)*

Divisive hierarchical clustering based on the notion of edge betweenness (Girvan-Newman Algorithm):

1. **Calculate edge betweenness:** find edges that are more "central" in the graph
2. **Delete high-betweenness edges:** delete the edges with highest betweenness
3. **Connected components** are communities

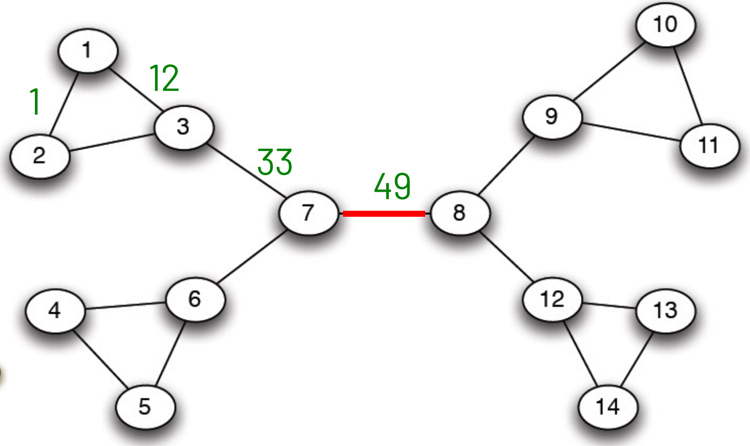
Repeat until stopping condition

Hierarchical decomposition!

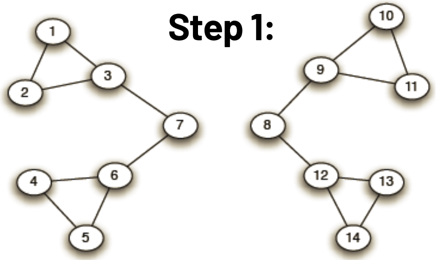


Community Detection

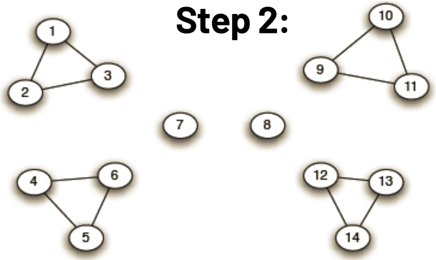
Divisive hierarchical clustering based on the notion of edge betweenness (Girvan-Newman Algorithm):



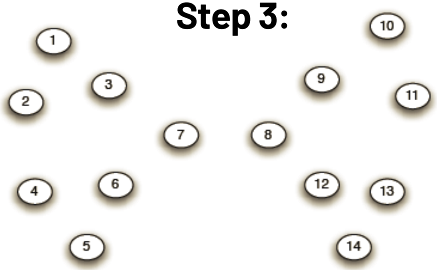
Step 1:



Step 2:

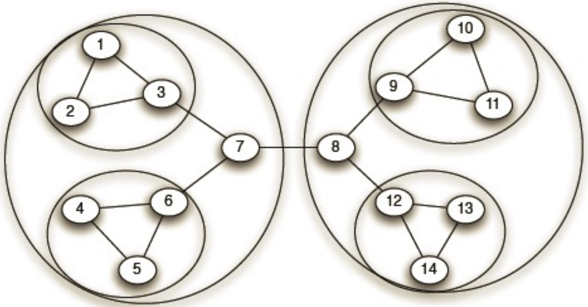


Step 3:



This method is not very effective for overlapping communities!

Hierarchical network decomposition:

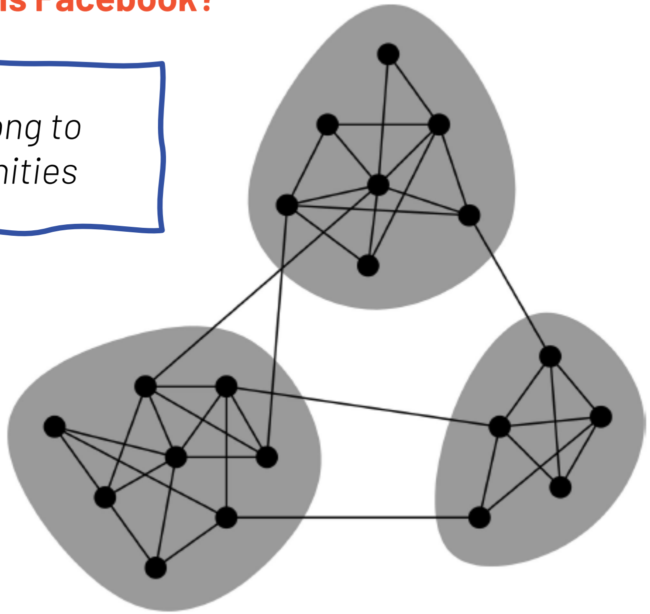


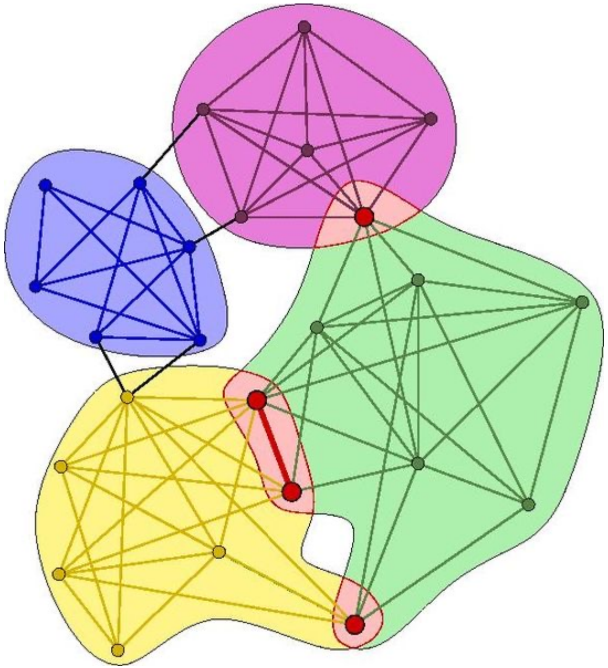
Non-overlapping vs. overlapping communities

See Community-Affiliation Graph in the Mining of Massive Datasets book

Which one is Facebook?

Some nodes belong to multiple communities





http://networksciencebook.com/chapter/9

Graph Data Analysis & Exploration

— Graph Exploration —

Matteo Lissandrini – Aalborg University

AALBORG UNIVERSITY

Outline

1. **Intro to Graph Exploration**
   - Taxonomy of Graph Exploration
   - Exploratory Search
   - Example Based Exploration
2. **Node-based Exploratory Search**
   - Seed-set expansion
   - Minimum Wiener Connector problem
   - Focused Clustering
   - Entity Set Search
3. **Structure-based Exploratory Search**
   - Reverse-engineering Queries
   - Entity Tuples
   - Exemplar Queries
   - Example-Based Graph suggestion

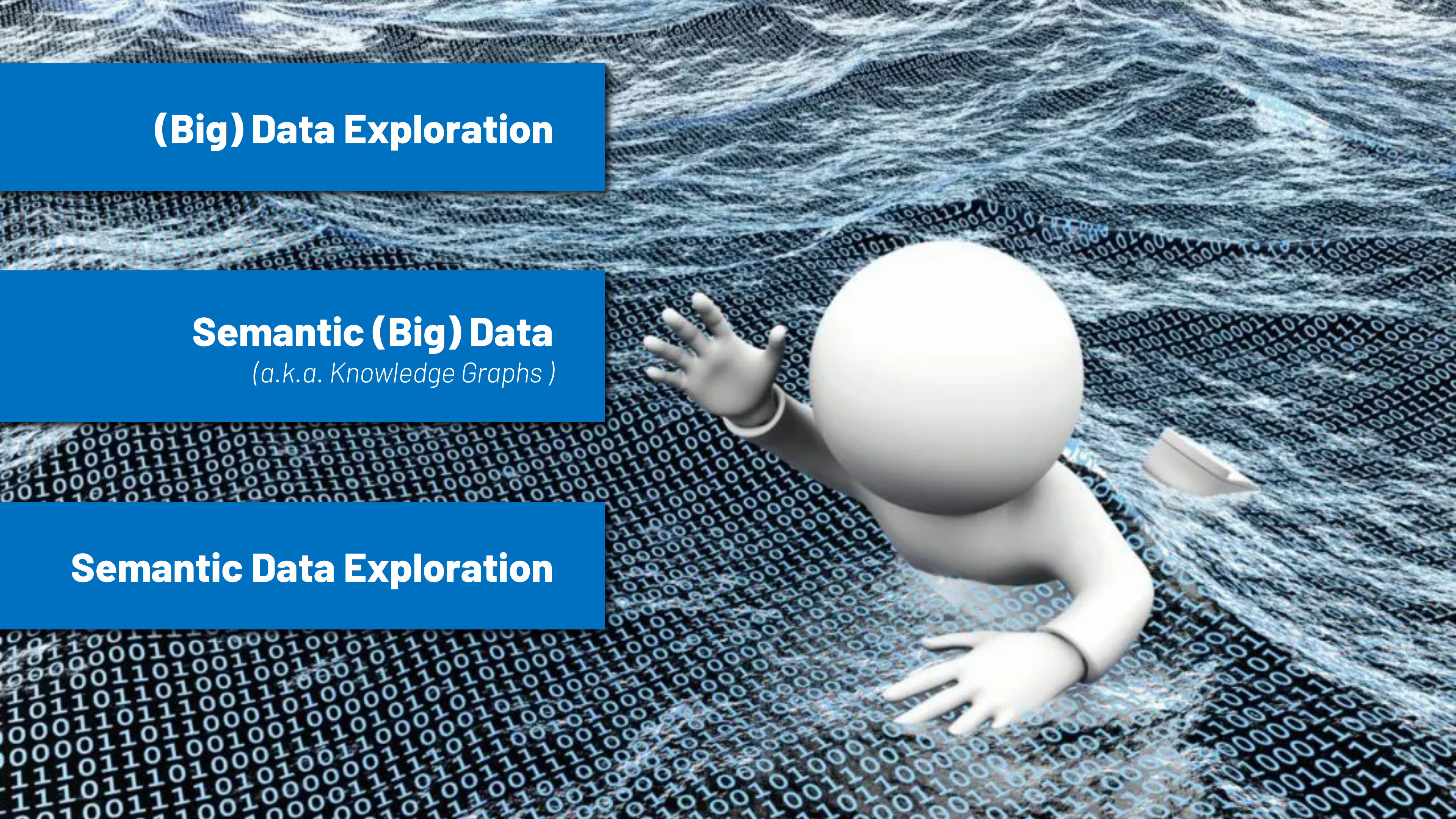

BIG DATA

(Big) Data Exploration

Semantic (Big) Data

(a.k.a. Knowledge Graphs)

Semantic Data Exploration

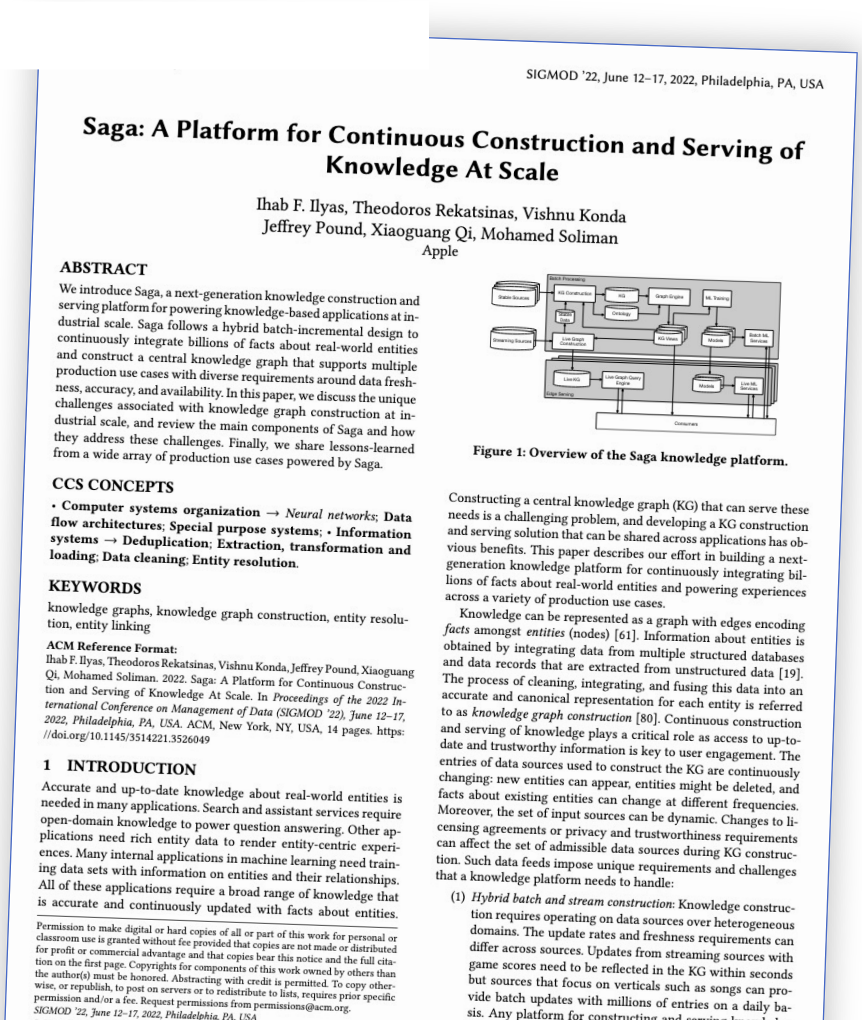


KG as a Data Model for Data Integration

The entries of data sources used to construct the KG **are continuously changing…**

[…]

Self-serve data onboarding: Low-effort **onboarding of new data sources** is important to ensure consistent growth of the KG.



SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA

Saga: A Platform for Continuous Construction and Serving of Knowledge At Scale

Ihab F. Ilyas, Theodoros Rekatsinas, Vishnu Konda
Jeffrey Pound, Xiaoguang Qi, Mohamed Soliman
Apple

ABSTRACT

We introduce Saga, a next-generation knowledge construction and serving platform for powering knowledge-based applications at industrial scale. Saga follows a hybrid batch-incremental design to continuously integrate billions of facts about real-world entities and construct a central knowledge graph that supports multiple production use cases with diverse requirements around data freshness, accuracy, and availability. In this paper, we discuss the unique challenges associated with knowledge graph construction at industrial scale, and review the main components of Saga and how they address these challenges. Finally, we share lessons-learned from a wide array of production use cases powered by Saga.

CCS CONCEPTS

• **Computer systems organization** → *Neural networks*; **Data flow architectures; Special purpose systems**; • **Information systems** → **Deduplication; Extraction, transformation and loading; Data cleaning; Entity resolution.**

KEYWORDS

knowledge graphs, knowledge graph construction, entity resolution, entity linking

ACM Reference Format:

Ihab F. Ilyas, Theodoros Rekatsinas, Vishnu Konda, Jeffrey Pound, Xiaoguang Qi, Mohamed Soliman. 2022. Saga: A Platform for Continuous Construction and Serving of Knowledge At Scale. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3514221.3526049

1 INTRODUCTION

Accurate and up-to-date knowledge about real-world entities is needed in many applications. Search and assistant services require open-domain knowledge to power question answering. Other applications need rich entity data to render entity-centric experiences. Many internal applications in machine learning need training data sets with information on entities and their relationships. All of these applications require a broad range of knowledge that is accurate and continuously updated with facts about entities.

Figure 1: Overview of the Saga knowledge platform.

Constructing a central knowledge graph (KG) that can serve these needs is a challenging problem, and developing a KG construction and serving solution that can be shared across applications has obvious benefits. This paper describes our effort in building a next-generation knowledge platform for continuously integrating billions of facts about real-world entities and powering experiences across a variety of production use cases.

Knowledge can be represented as a graph with edges encoding *facts* amongst *entities* (nodes) [61]. Information about entities is obtained by integrating data from multiple structured databases and data records that are extracted from unstructured data [19]. The process of cleaning, integrating, and fusing this data into an accurate and canonical representation for each entity is referred to as *knowledge graph construction* [80]. Continuous construction and serving of knowledge plays a critical role as access to up-to-date and trustworthy information is key to user engagement. The entries of data sources used to construct the KG are continuously changing: new entities can appear, entities might be deleted, and facts about existing entities can change at different frequencies. Moreover, the set of input sources can be dynamic. Changes to licensing agreements or privacy and trustworthiness requirements can affect the set of admissible data sources during KG construction. Such data feeds impose unique requirements and challenges that a knowledge platform needs to handle:

(1) *Hybrid batch and stream construction*: Knowledge construction requires operating on data sources over heterogeneous domains. The update rates and freshness requirements can differ across sources. Updates from streaming sources with game scores need to be reflected in the KG within seconds but sources that focus on verticals such as songs can provide batch updates with millions of entries on a daily basis. Any platform for constructing and serving knowl…

Modern Information Search Use-case

Are there any other fraudulent cases like this Jon Doe’s bill

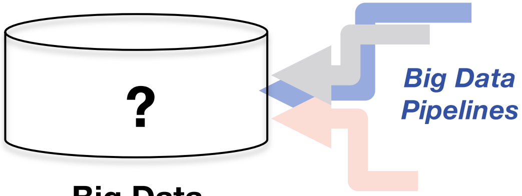
Challenges

1) Not sure **about the data** we have

2) Not clear **what we are looking for**

?

How do we describe what we are looking for?

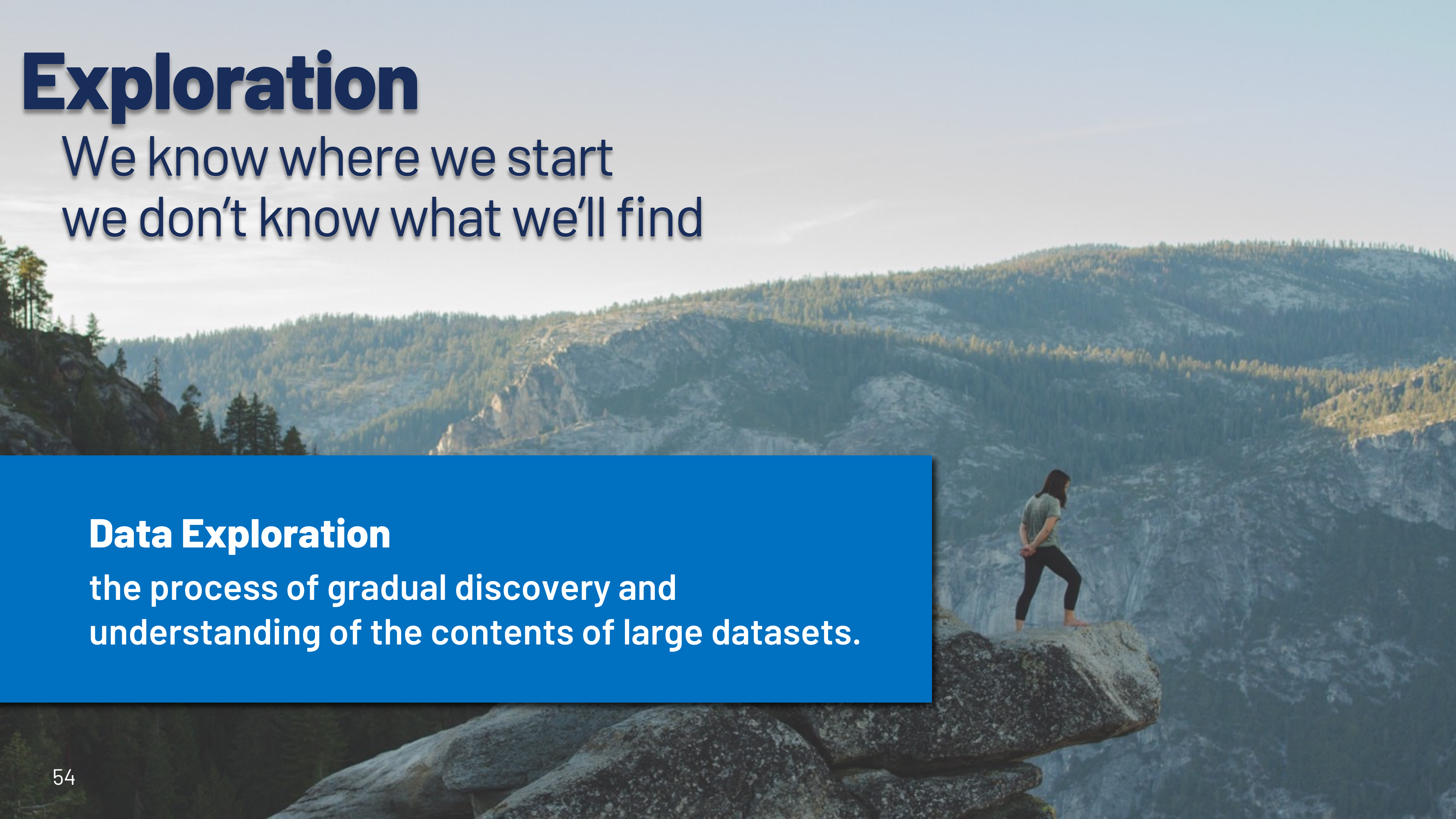


Big Data

what are the contents
of our database?

The Data Novice:
A user unfamiliar with the data at hand and its structure

Opportunity: Empower Data Scientists to find the information they need in large heterogenous data repositories



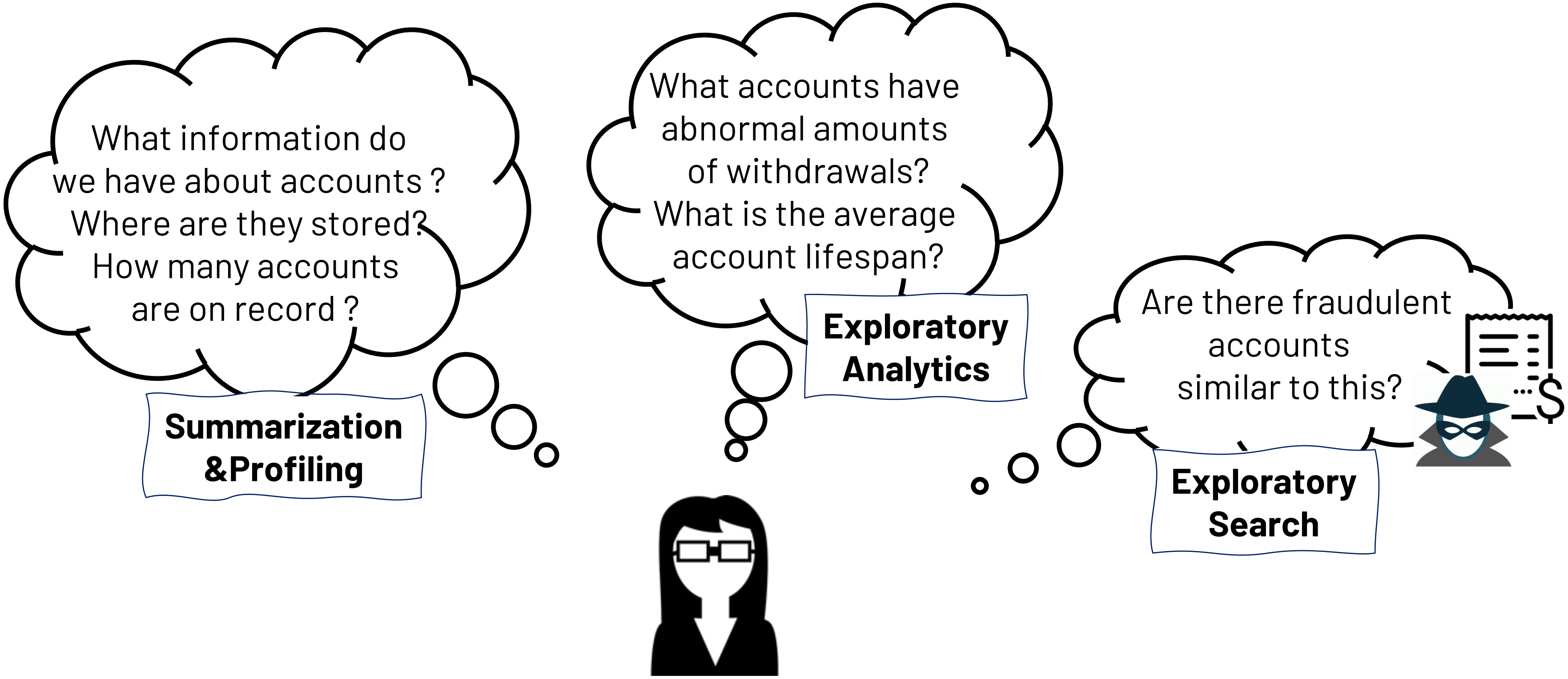
Exploration

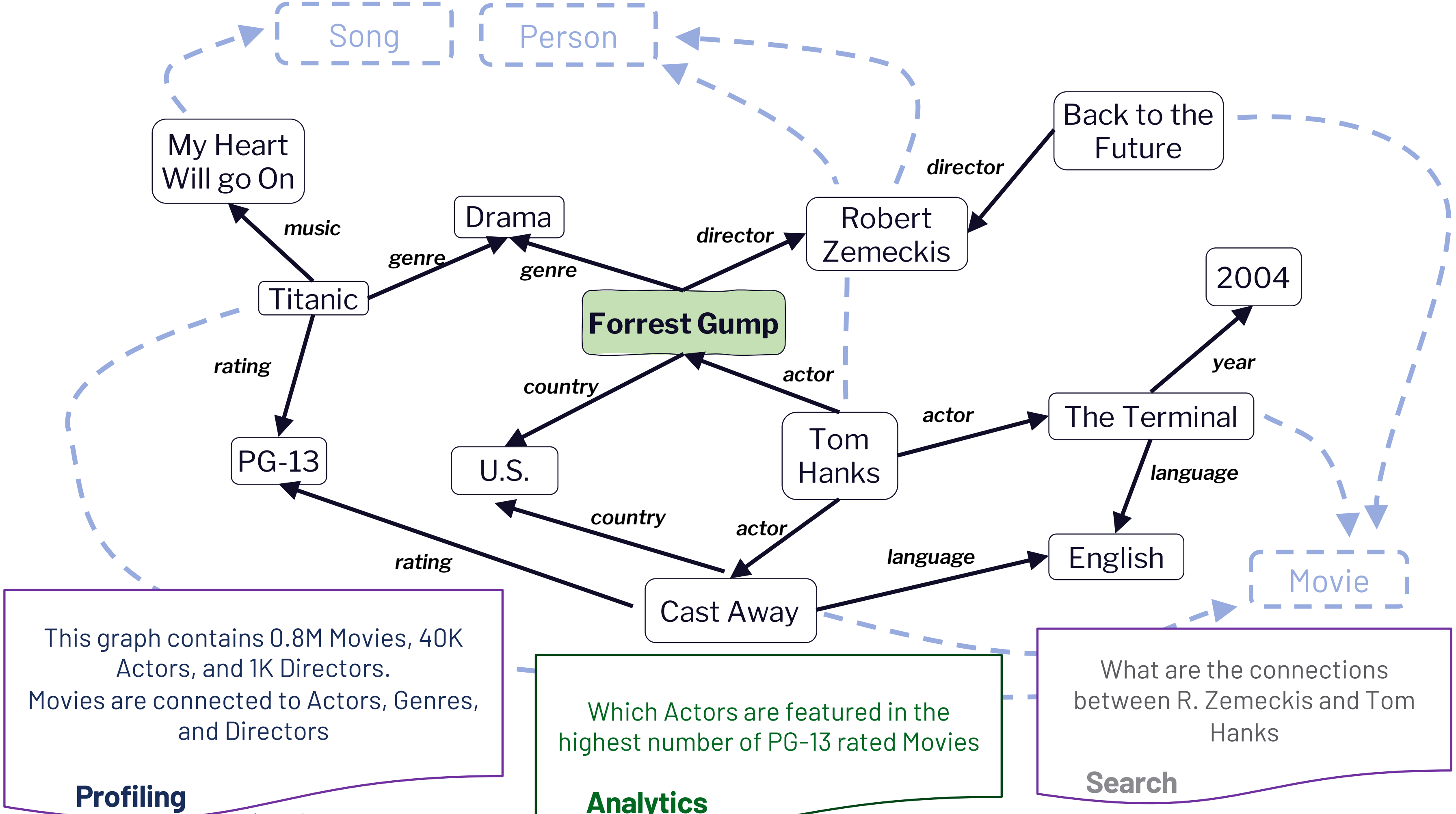
We know where we start
we don't know what we'll find

Data Exploration

the process of gradual discovery and understanding of the contents of large datasets.

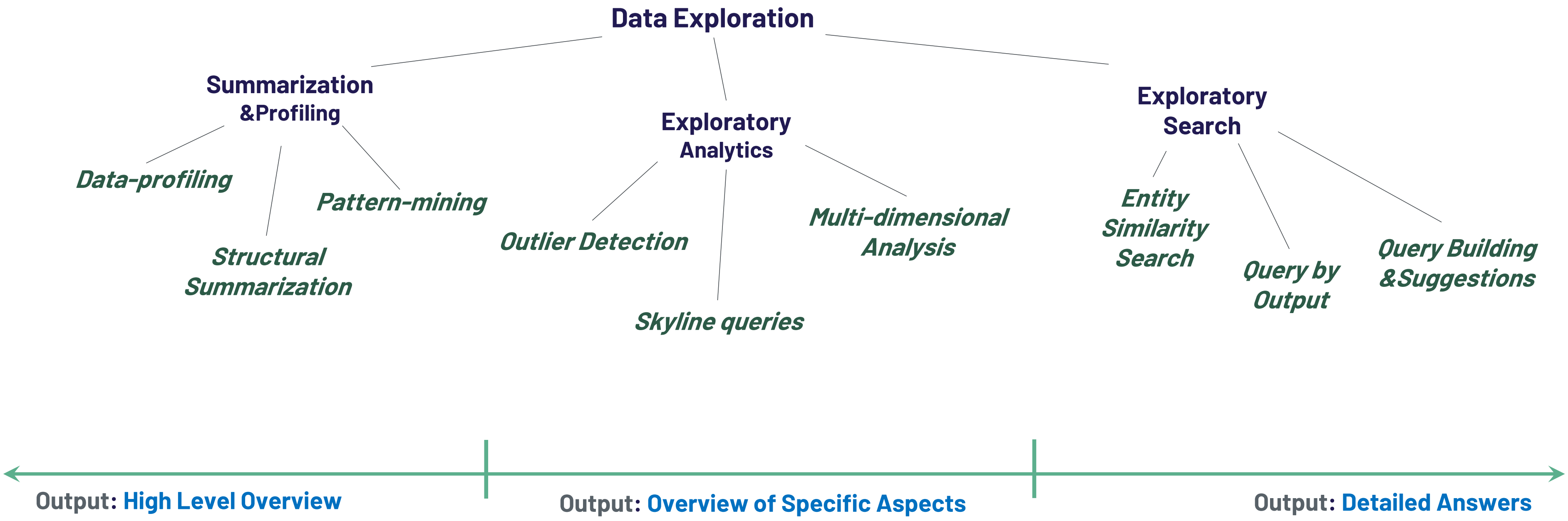
Data Exploration Needs





https://data-exploration.ml

Data Exploration Methods



Data Exploration Methods

Data Exploration

- **Summarization &Profiling**
  - *Data-profiling*
  - *Pattern-mining*
  - *Structural Summarization*
- **Exploratory Analytics**
  - *Outlier Detection*
  - *Skyline queries*
  - *Multi-dimensional Analysis*
- **Exploratory Search**
  - *Entity Similarity Search*
  - *Query by Output*
  - *Query Building &Suggestions*

No Interaction and Personalization	Medium-High Interactivity	High Interactivity and Personalization
Requires: No Domain Knowledge	Requires: High-level Information Need	Requires: Detailed Sample or Query Intent
Output: High Level Overview	Output: Overview of Specific Aspects	Output: Detailed Answers

KG Profiling

Heiko Paulheim [2017]
http://ceur-ws.org/Vol-1927/paper1.pdf

Obtain a basic understanding of the contents of a KG

1. How many instances? How many classes?
2. What's the vocabulary (predicates/attributes)
3. Are there big-hubs? Are there disconnected islands?

Table 1: Global Properties of the Knowledge Graphs compared in this paper

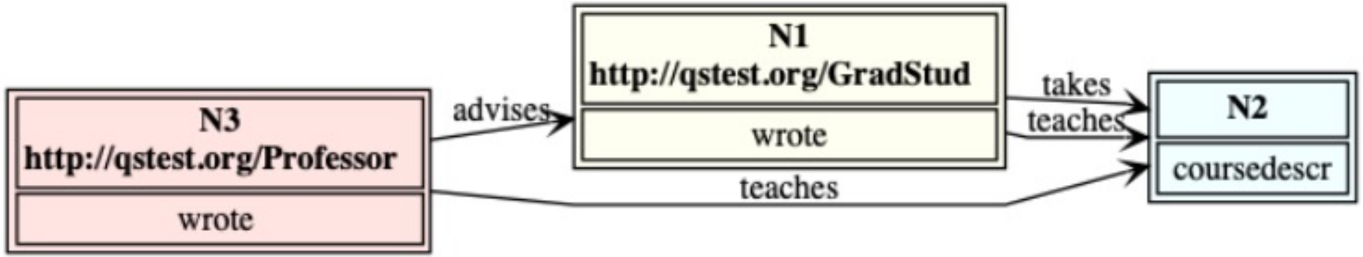
	DBpedia	YAGO	Wikidata	OpenCyc	NELL
Version	2016-04	YAGO3	2016-08-01	2016-09-05	08m.995
# instances	5,109,890	5,130,031	17,581,152	118,125	1,974,297
# axioms	397,831,457	1,435,808,056	1,633,309,138	2,413,894	3,402,971
avg. indegree	13.52	17.44	9.83	10.03	5.33
avg. outdegree	47.55	101.86	41.25	9.23	1.25
# classes	754	576,331	30,765	116,822	290
# relations	3,555	93,659	11,053	165	1,334
Releases	biyearly	> 1 year	live	> 1 year	1-2 days

KG Summarization & Pattern Mining

Surveys:
Kellou-Menouer et al [2022]
Čebirić et al [2019]

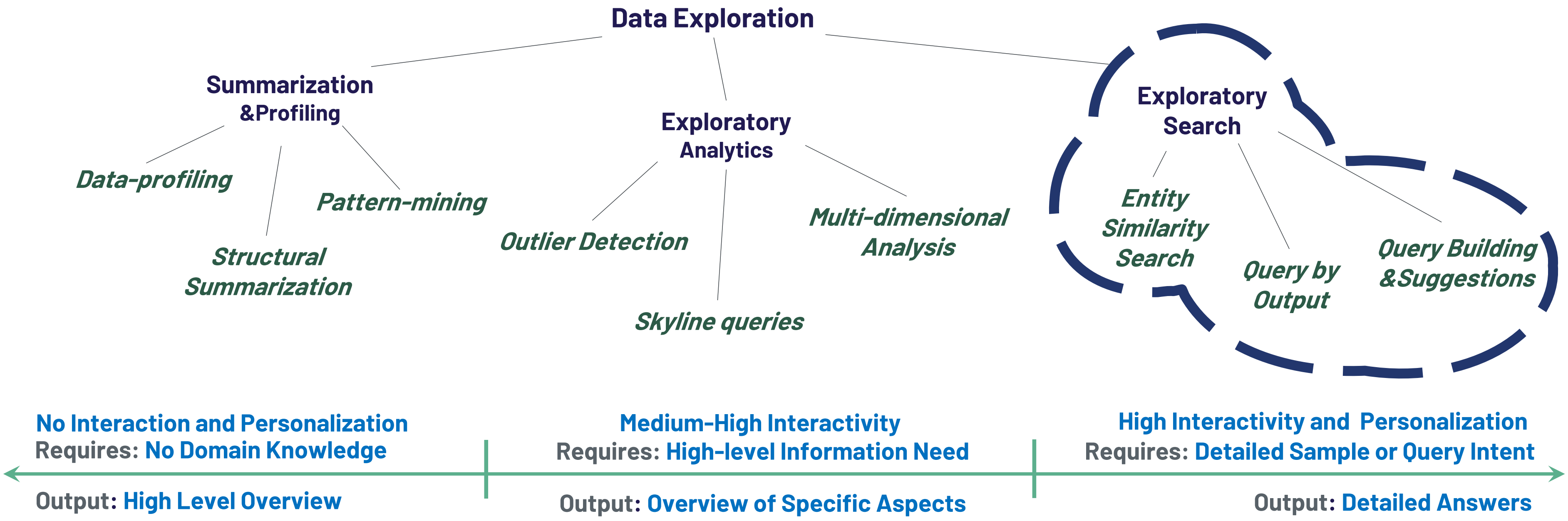
Extract overall structural information

1. How are classes connected?
2. Which predicates and attributes are shared by entities of this type?
3. What is the prevalence of connections across nodes with this properties?

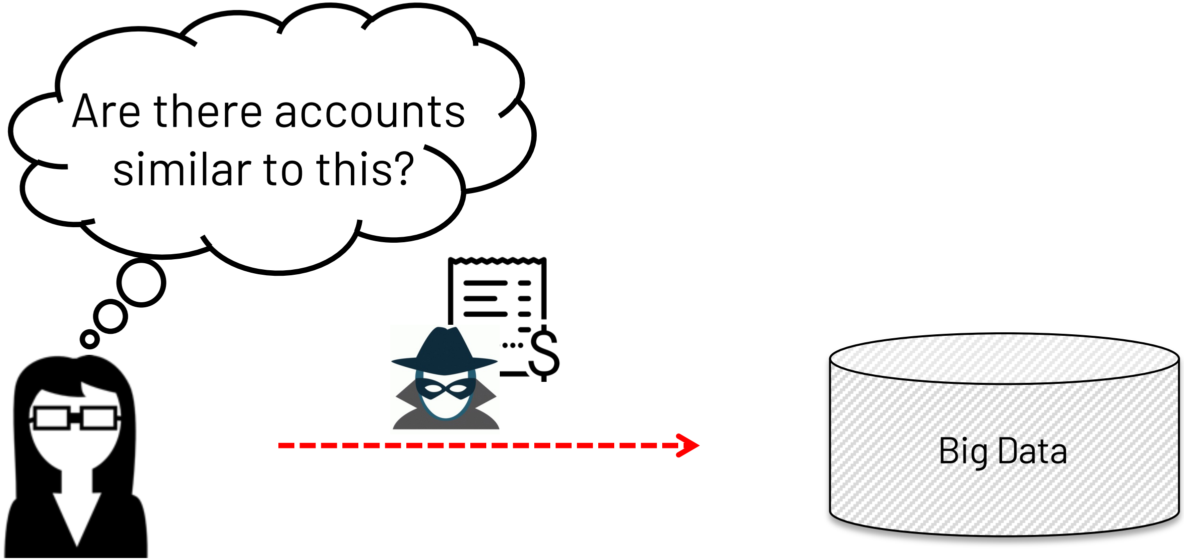


https://link.springer.com/article/10.1007/s00778-021-00717-x
https://link.springer.com/article/10.1007/s00778-018-0528-3

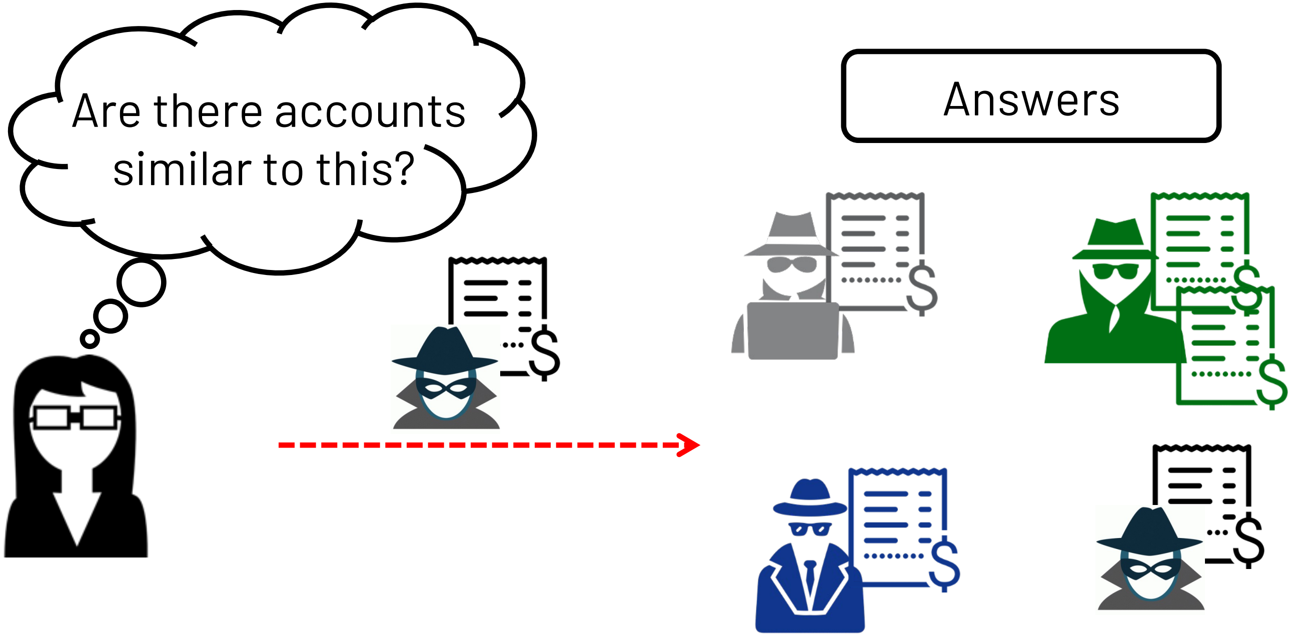
Data Exploration Methods



Examples as Exploratory Methods



Examples as Exploratory Methods



Example is always more efficacious than precept
Samuel Johnson, Rasselas (1759)

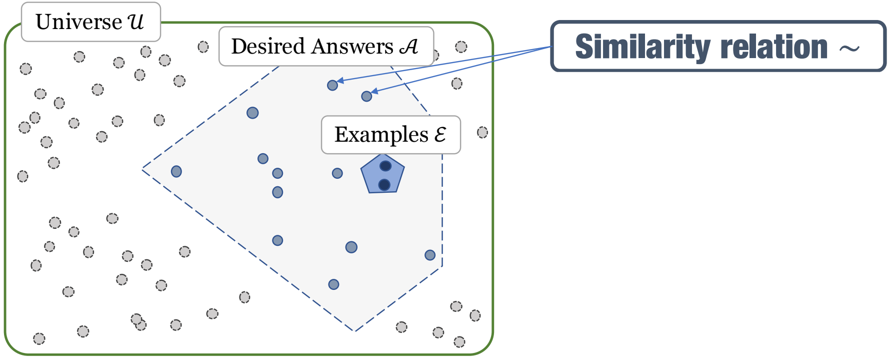
Similarities are the key . . .

If we knew how similar each item is with respect to any other for each user, we would know the answer

The Example-based problem

Given

a set of examples $\mathcal{E}$ from a universe $\mathcal{U}$



Find

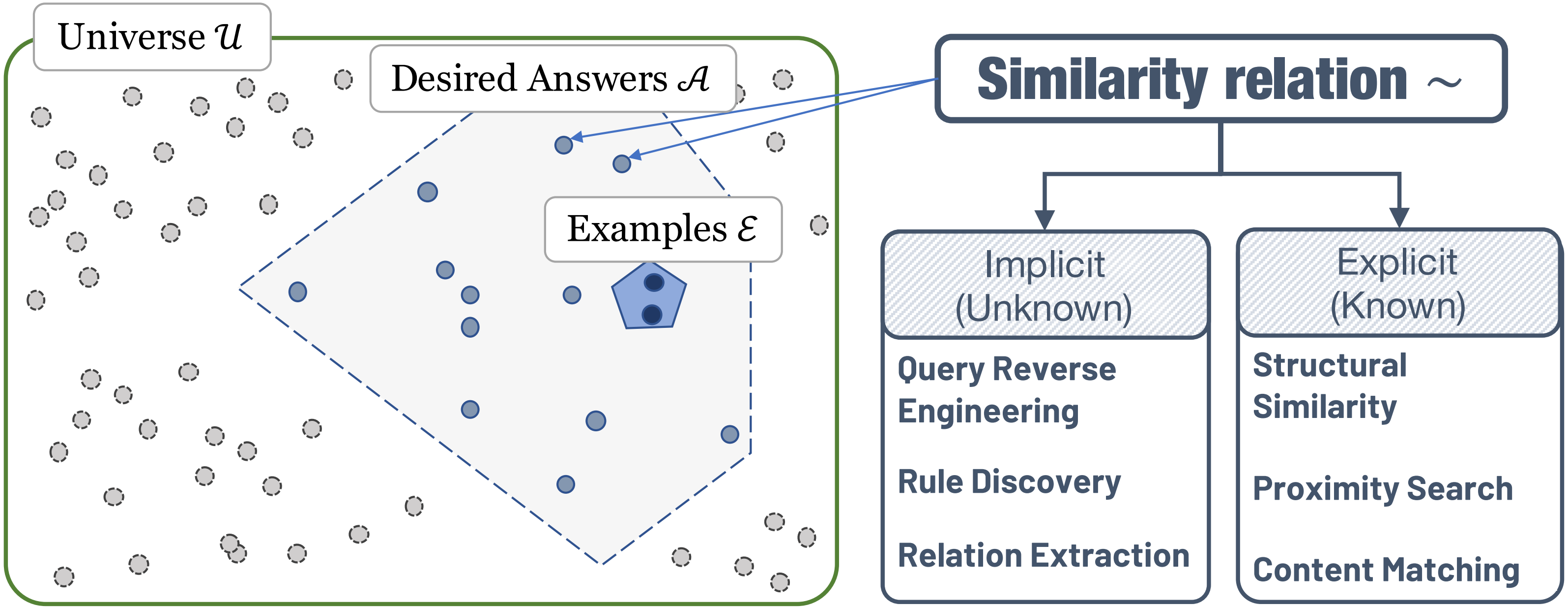
a similarity " $\sim$ "

such that

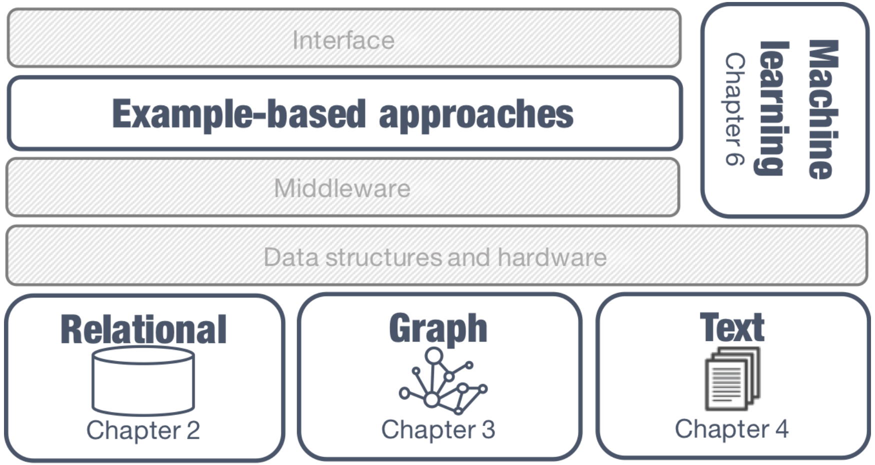
1. When $\mathcal{E}$ is part of the answers $\mathcal{A}$ (partially or totally)
2. The answers in $\mathcal{A}$ are the most similar to the examples in $\mathcal{E}$ according to " $\sim$ "

What similarity " $\sim$ " should we use ?
How do we identify " $\sim$ " (for each user) ?

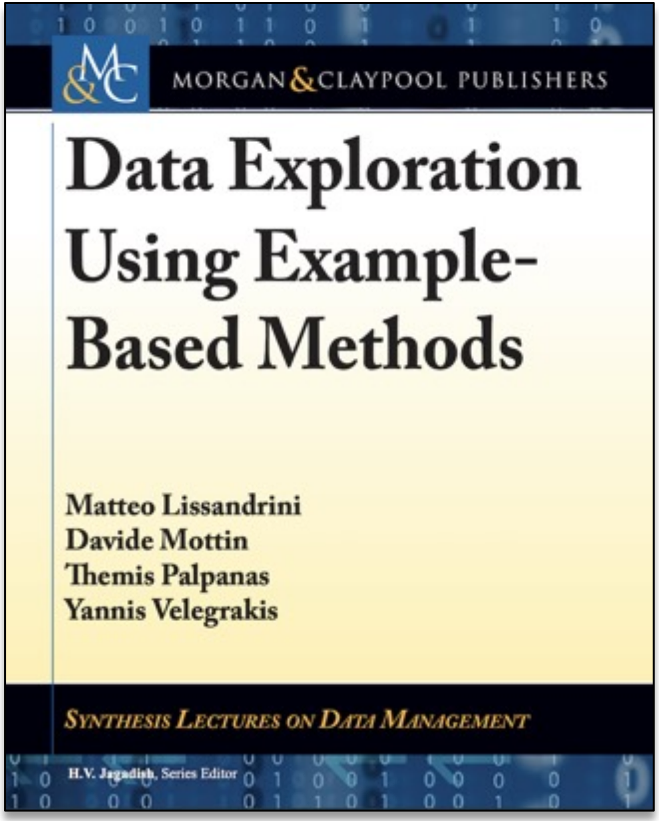
Example-based methods



Book on Example-based methods



Slides and Materials
https://data-exploration.ml/



Exemplar Queries

Example-driven graph search

Mottin et al. [2014,2016]

Input: Q_e, an example element of interest

Output: set of elements in the desired result set

Nodes/Entities

Edges/Facts

Structures

Exemplar Query Evaluation

- evaluate Q_e in a database D, finding a sample S
- find the set of elements A similar to S given a similarity relation
- [OPTIONAL] return only the subset A^R that are relevant

Usually requires an intermediate step:
User input (keywords) → Element in the graph

SIMILARITY for GRAPHS

SEARCHING FOR: **Nodes** | **Structures**

BY LOOKING AT: Proximity | Properties | Queries | Relationships

PRODUCES:

Nodes

- **Proximity**
  - Mediator Nodes [Gionis et al. '15 Ruchansky et al.'15]
  - Clusters [Perozzi et al.'14, Kloumann et al. 14]
- **Properties**
  - Similar Entities [Metzger et al.'13, Sobczak et al.'15]

Structures

- **Queries**
  - Path Queries [Bonifati et al.'15]
  - SPARQL [Arenas et al.'16]
- **Relationships**
  - Entity Tuples [Jayaram et al.'15]
  - Similar Structures [Mottin et al.'14 & '16, Xie et al.'17, Lissandrini et al. '18 & '20]

Challenges: 1. Discover User Preference

2. Efficient Search

SIMILARITY for GRAPHS

SEARCHING FOR

BY LOOKING AT

PRODUCES

Nodes

Proximity

Mediator Nodes
[Gionis et al. '15
Ruchansky et al.'15]

Clusters
[Perozzi et al.'14,
Kloumann et al. 14]

Properties

Similar Entities
[Metzger et al.'13,
Sobczak et al.'15]

Structures

Queries

Path Queries
[Bonifati et al.'15]

SPARQL
[Arenas et al.'16]

Relationships

Entity Tuples
[Jayaram et al.'15]

Similar Structures
[Mottin et al.'14 & '16,
Xie et al.'17,
Lissandrini et al. '18 & '20]

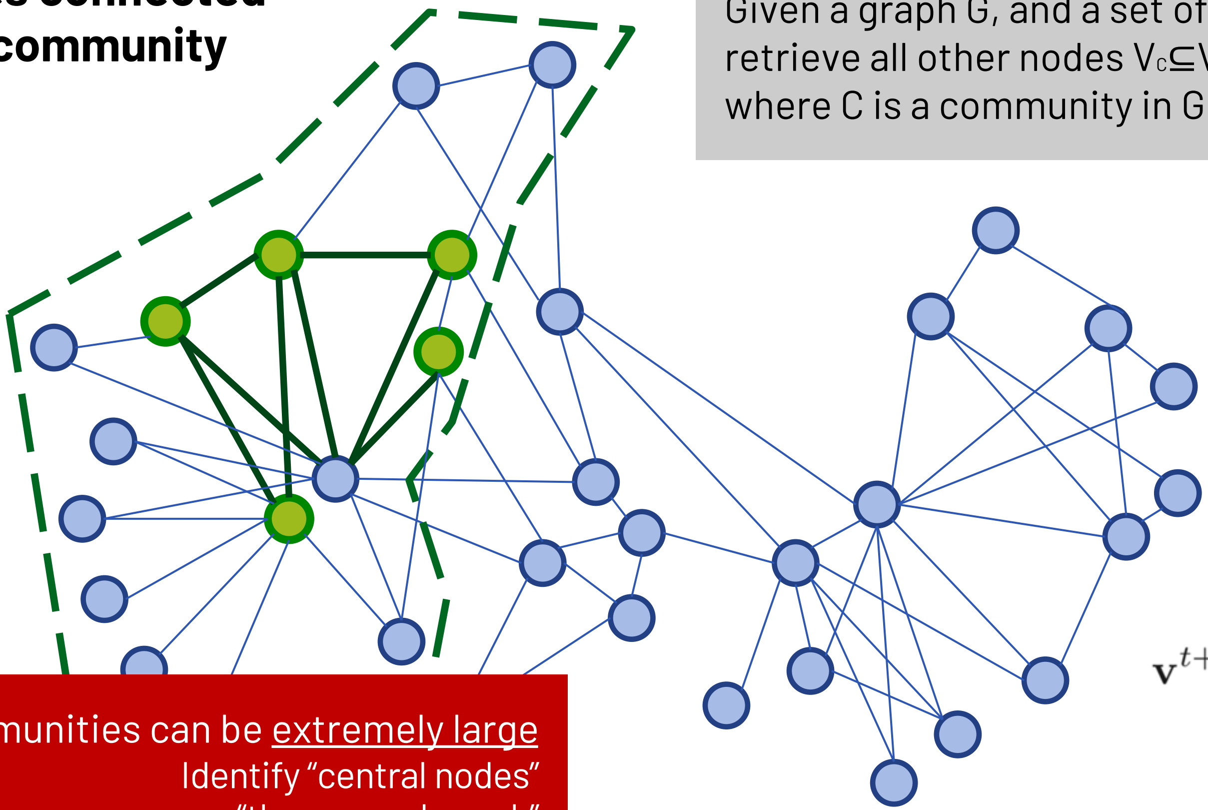
Challenges: 1. Discover User Preference
2. Efficient Search

Seed Set Expansion

Nodes connected by a community

Kloumann and Kleinberg [2014]

Given a graph G, and a set of query nodes $V_Q \subseteq V_G$, retrieve all other nodes $V_C \subseteq V_G$, where C is a community in G, and $V_Q \subseteq V_C$.



Solution: PPR

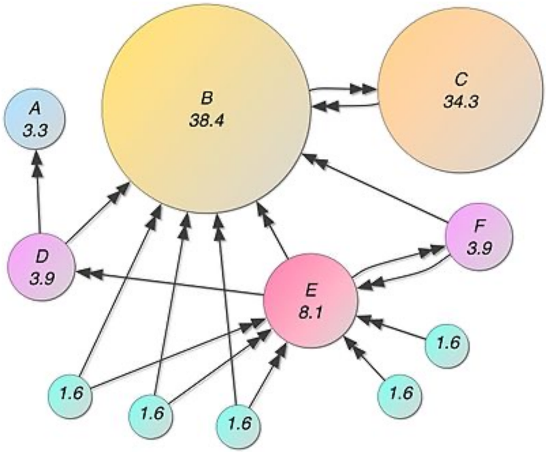
$$\mathbf{v}^{t+1} = (1-\alpha)\mathbf{M}\cdot\mathbf{v}^t + \alpha\mathbf{v}^0$$

Communities can be <u>extremely large</u>
Identify “central nodes”
or “the core subgraph”

Traverse (Document) Networks

How to navigate links and connections

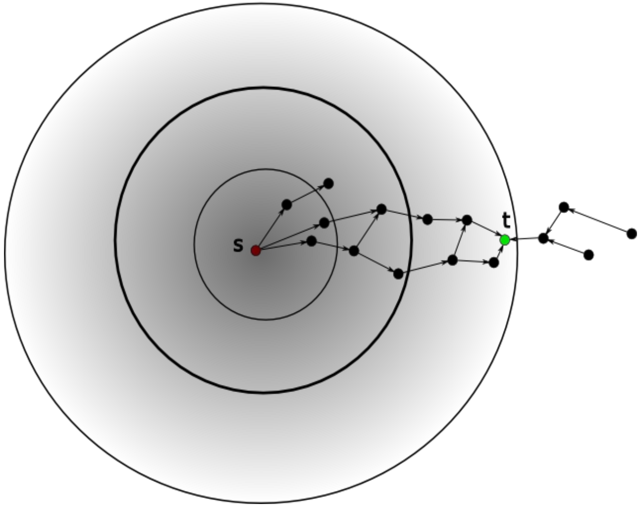
El-Arini and Guestrin [2011]
Jia and Saule [2017]



Personalized Page Rank

- Start from seed nodes, i.e. the documents D_{rel}
- Navigate towards locally connected nodes

Example based Exploration implies locality



Global Page Rank

Starting from a random node, traversing randomly, random restart point anywhere in the graph

CHALLENGE:
Identify meaningful transition probabilities

E.g., El-Arini and Guestrin [2011]

Personalized Page Rank

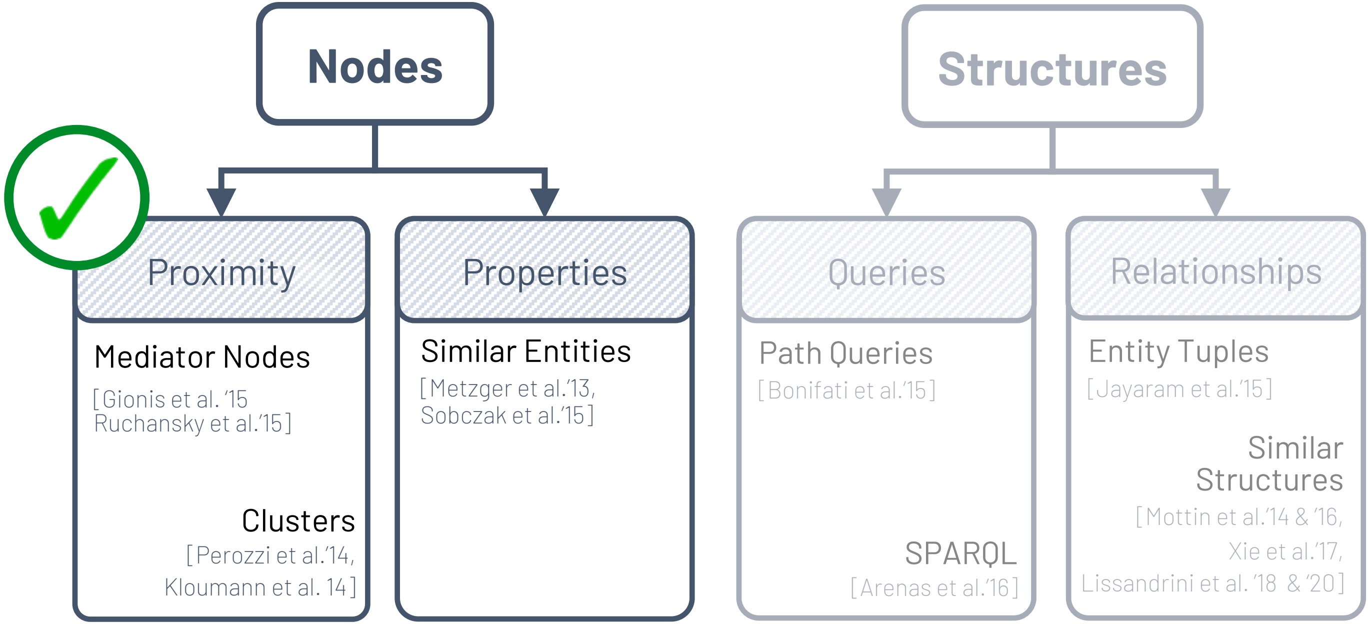
Starting from a limited set of nodes, traversing randomly, restart point is one in the initial set.
<u>Bound not to travel too far</u>

SIMILARITY for GRAPHS

SEARCHING FOR

BY LOOKING AT

PRODUCES



iQBEES: Entity Search by Example

Knowledge Graph Search

Metzger et al. [2013]
Sobczak et al. [2015]

Model: Knowledge Graph (Edge-labels)

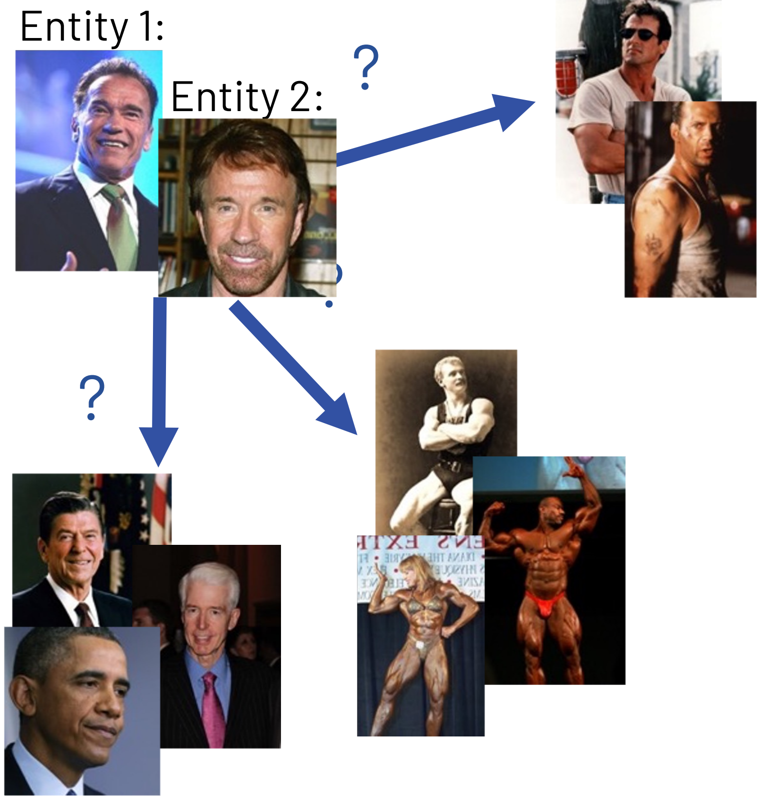
Query: A set of Entities Q

Similarity: Shared semantic properties

Output: A Set of <u>Similar Entities</u> (ranked)

Case: Products ⟶ Products with similar aspects

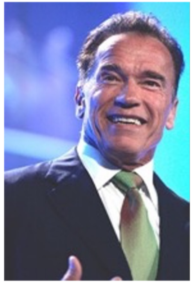
Case: Social Media ⟶ User recommendation



Maximal Aspect Sets

Selecting Features of Entity Similarity

Metzger et al. [2013]
Sobczak et al. [2015]



?x sport BodyBuilding

?x type AmericanActor

Is not maximal if
Adding any aspect
→ E(A)={Arnold}

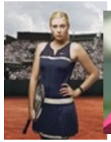
?x type AmericanActor

?x governorOf California

Include
Typical Types

?x hasHeight 1.88m

?x type Entity



Use most
Specific Type

?x type AmericanActor

?x actedIn TheExpendables

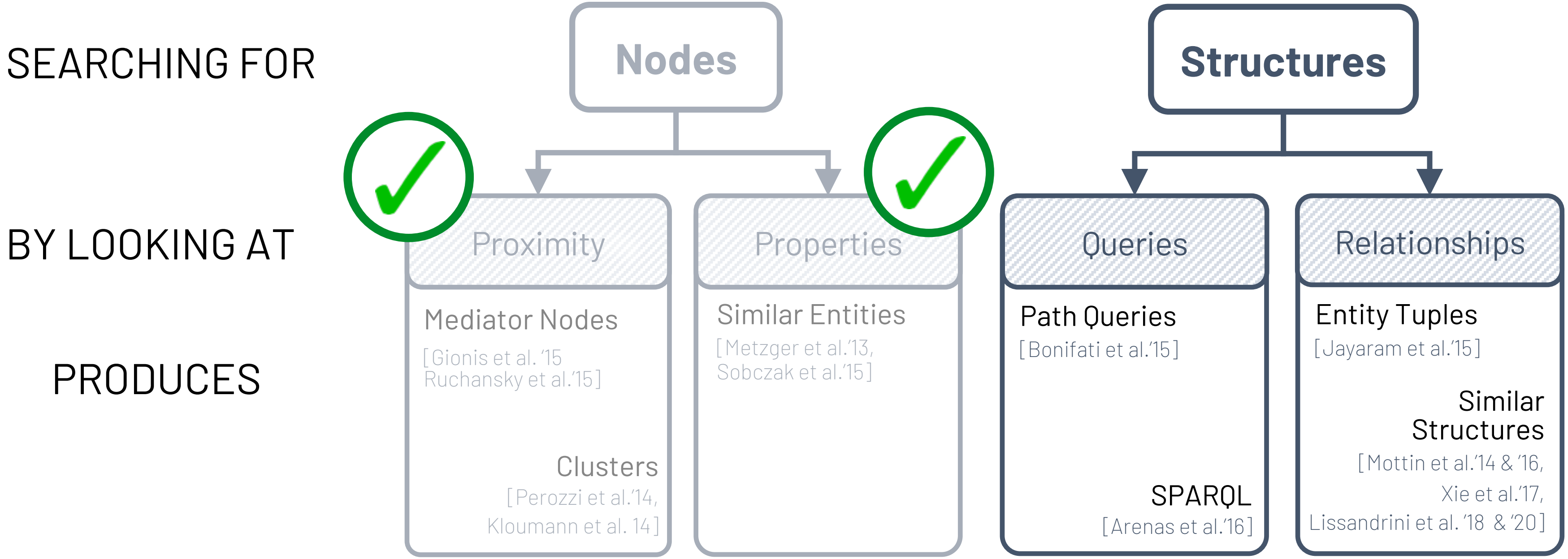
?x type ActionActor

REPEATABLE
Update Q

1. Prune generic aspects

2. Rank Set of aspects

SIMILARITY for GRAPHS



Reverse engineering SPARQL queries

Arenas et al. [2016]

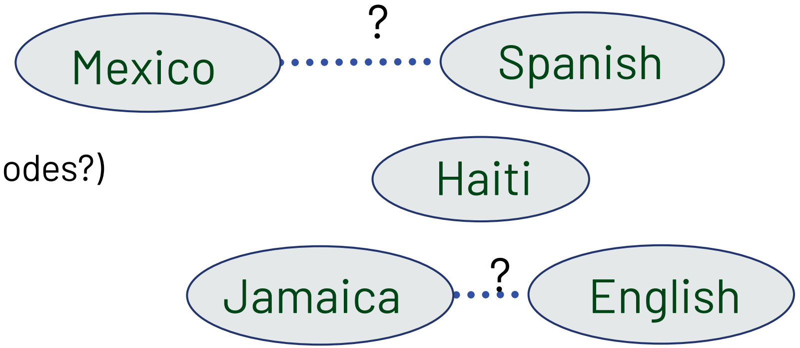
Knowledge Graph Search

Model: Knowledge Graph (Edge-labels)

Query: Set of Answers → Not Graphs but Tuples (of Nodes?)

Similarity: common AND/OPT/FILTER query

Output: a SPARQL query / query results



	?e1	?e2
M1	**Mexico**	**Spanish**
M2	**Haiti**	
M3	**Jamaica**	**English**

```
MATCH (?X, is_a, Country)
OPT (?X, has_language, ?Y)
```

Case: Open Data ⟶ Query Unknown Schema

Case: Novice User ⟶ Avoid SPARQL

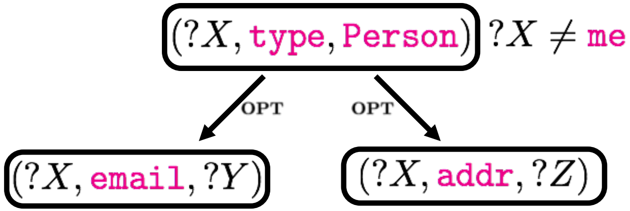
Reverse engineering SPARQL queries

Arenas et al. [2016]

Challenges and Complexity

Query: Set of Variable Mappings

	$?X$	$?Y$	$?Z$
M1	John		
M2	Mary	mary@email.eu	
M3	Lucy		Roses Street



Incomplete Mappings are treated as OPTIONAL
Typical of RDF queries

Enumerate all possible SPARQL queries satisfied by the mappings

INTRACTABLE

Σ_2^p–complete

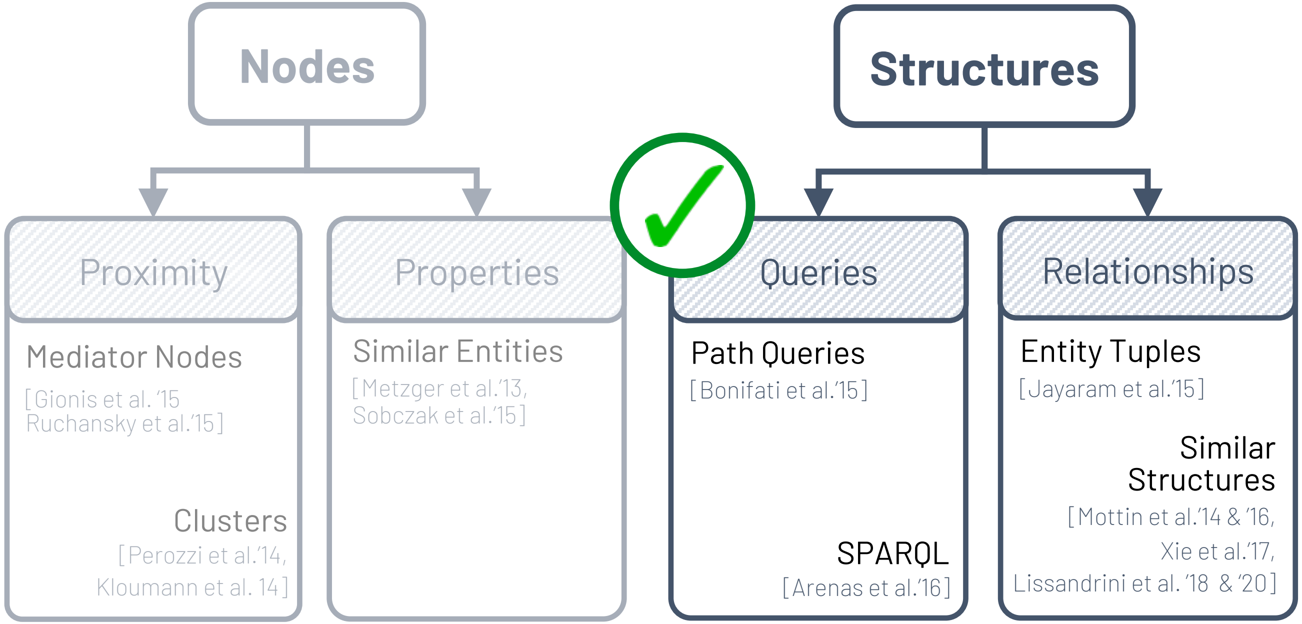
Build tree-shaped SPARQL queries IMPLIED by the mappings

SIMILARITY for GRAPHS

SEARCHING FOR

BY LOOKING AT

PRODUCES



Graph Exemplar Queries

Search for Structures

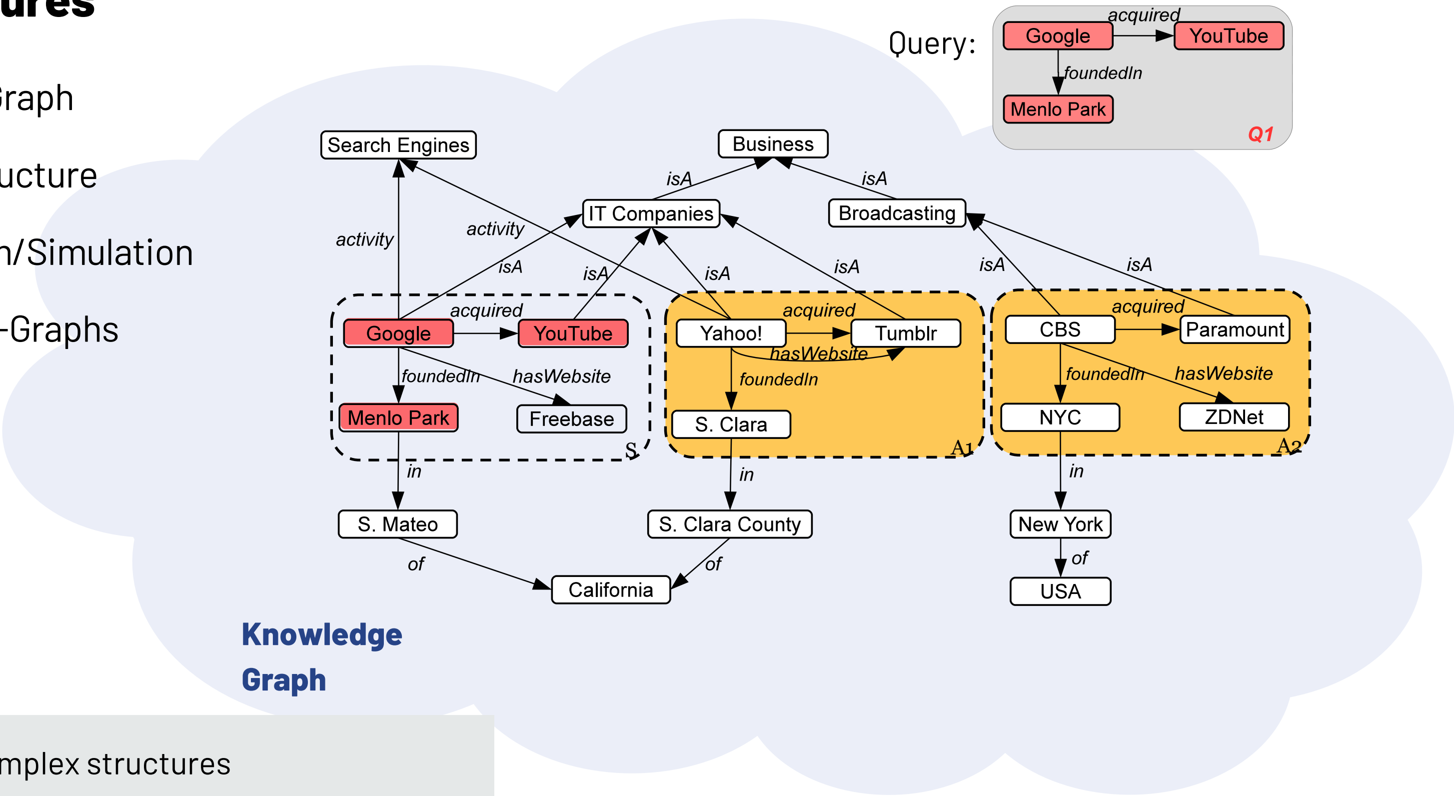
Mottin et al. [2016]

Model: Knowledge Graph

Query: Example Structure

Similarity: Isomorphism/Simulation

Output: A set of Sub-Graphs

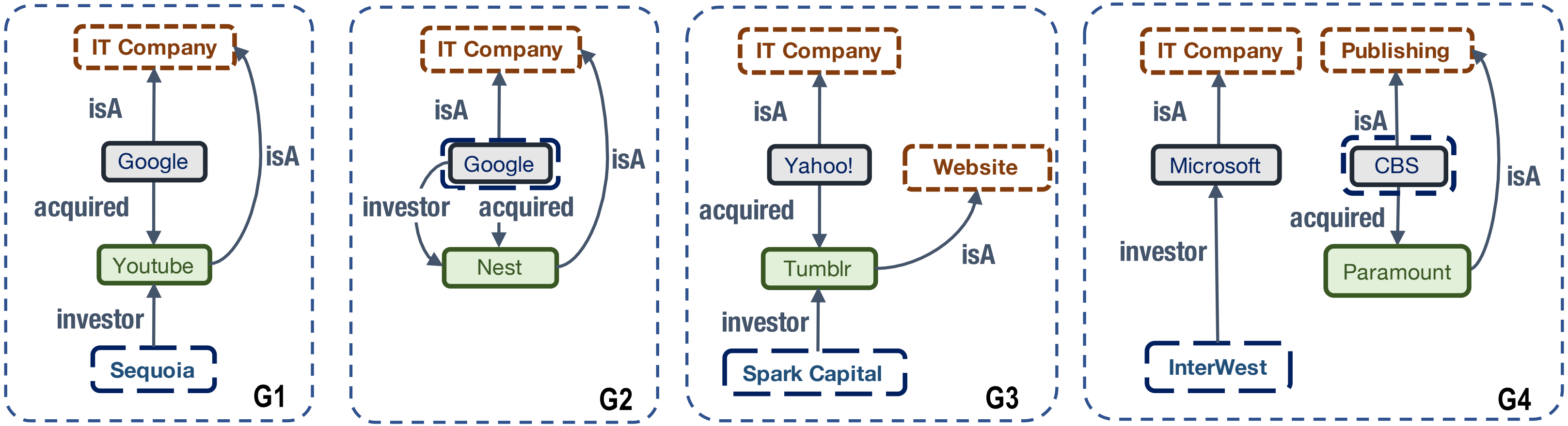


Case: Rich Schema ⟶ Find complex structures

Graph Isomorphism vs. Simulation Variants

Structural Congruence/Similarity

Isomorphism requires an bijective **function**
Simulation requires only a **parent-child** edge preserving **relation**
Strong Simulation requires also **child-parent**, **connectivity** and **limited diameter**

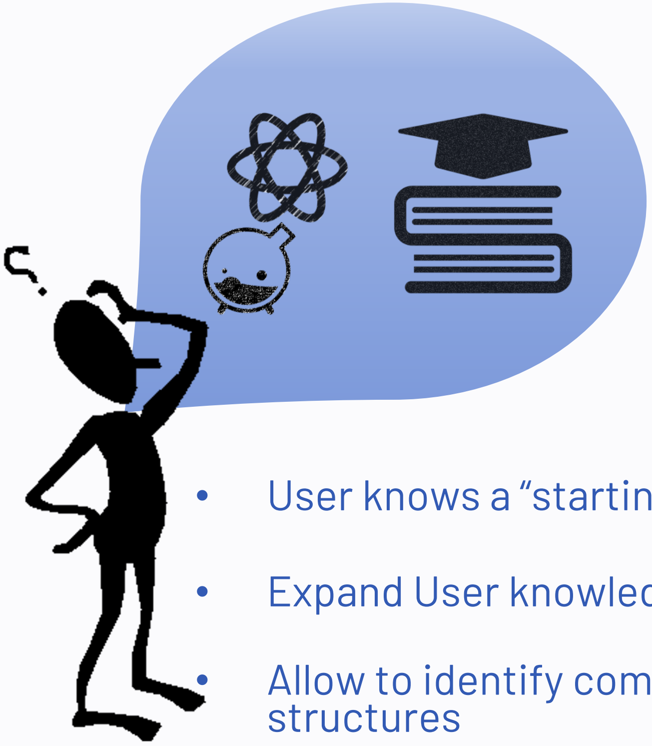


Example of Simulating (G1~ {G2,G3,G4}) and Strong-simulating Graphs (G1≈G2)

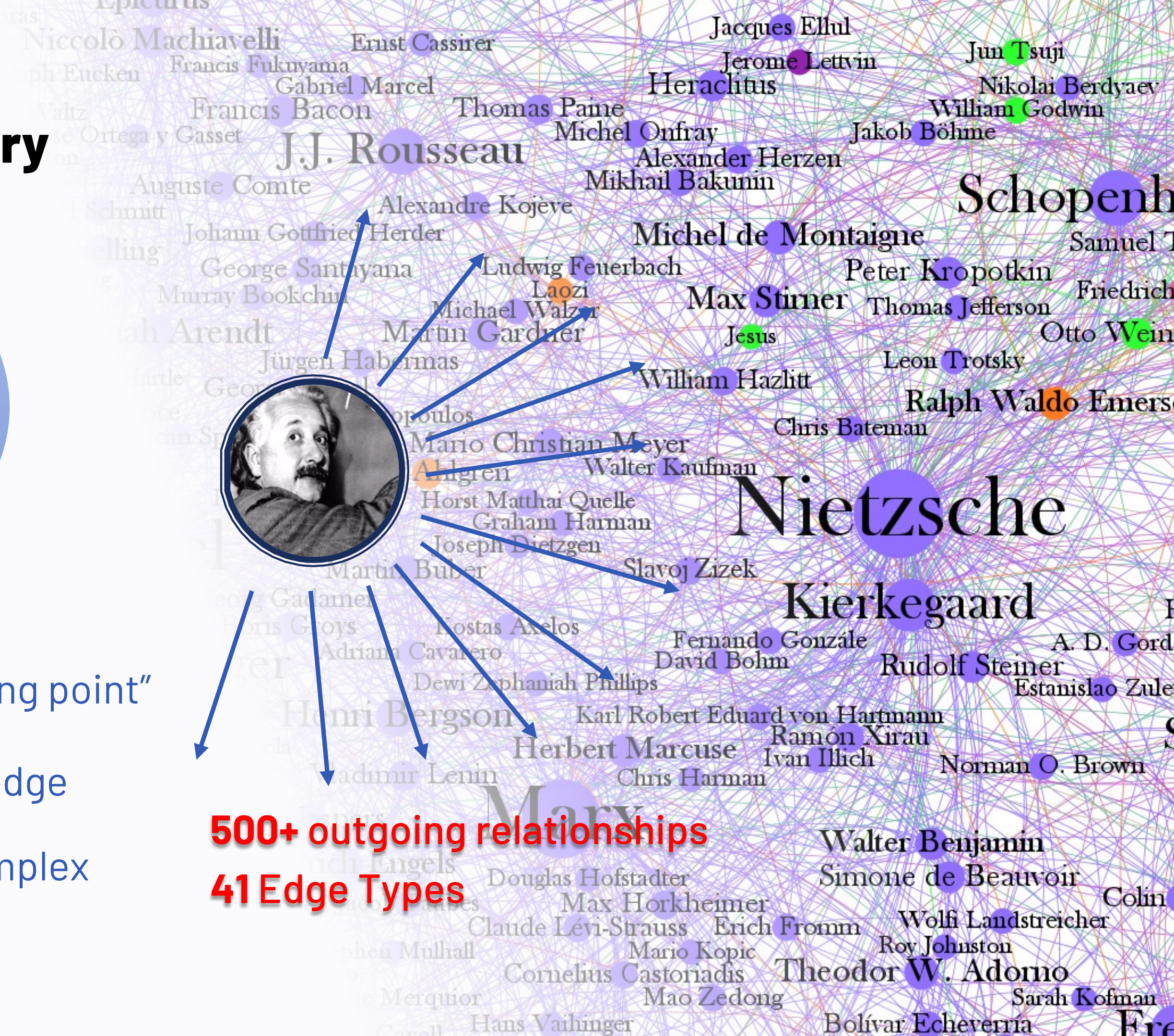
Strong Simulation preserves close connectivity

Strong simulation: Capturing topology in graph pattern matching
– Shuai Ma et al., 2014

Help the user formulate an Exploratory Graph Query

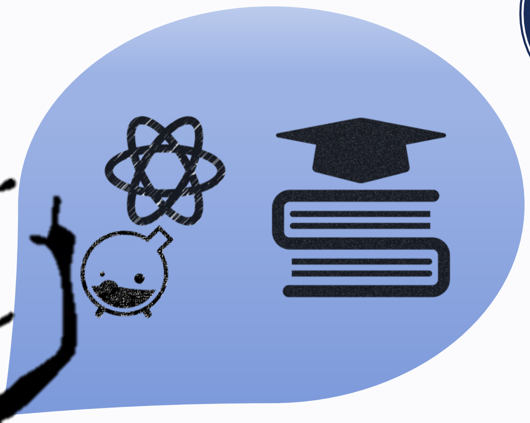


- User knows a "starting point"
- Expand User knowledge
- Allow to identify complex structures

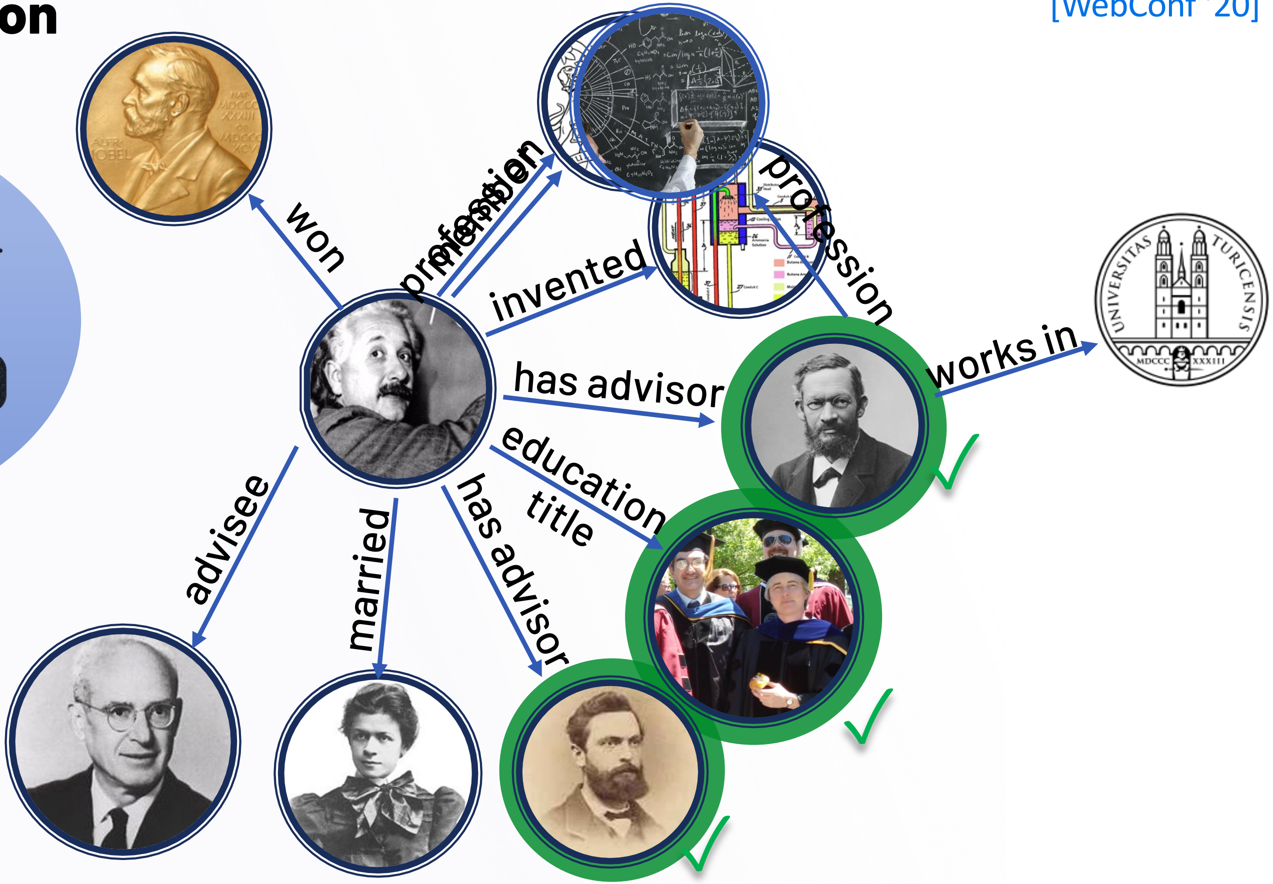


Graph-Query Suggestion

Lissandrini et al.
[WebConf ‘20]



User Interaction
- Narrow the options
- Select suggestion
- Focused Expansions

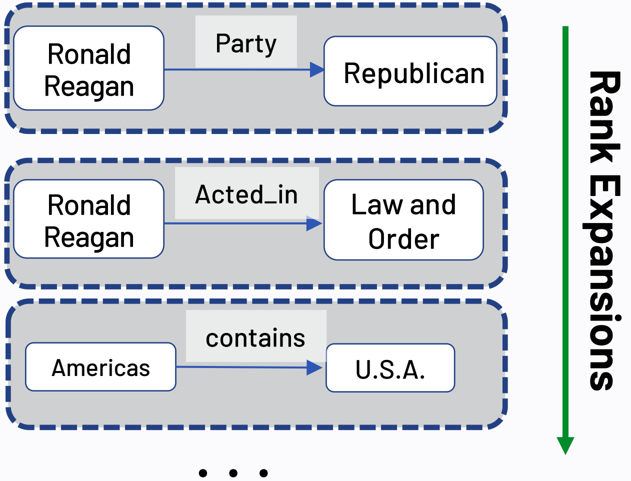


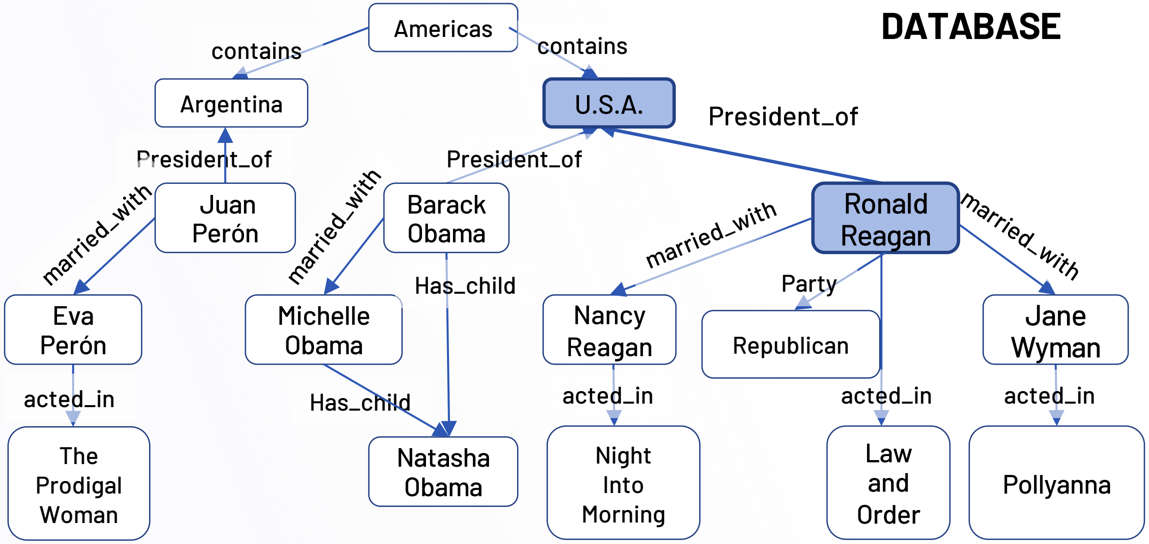
Suggesting Query Expansions

Lissandrini et al.
[WebConf '20]

The User Search

Which Expansion to Suggest?



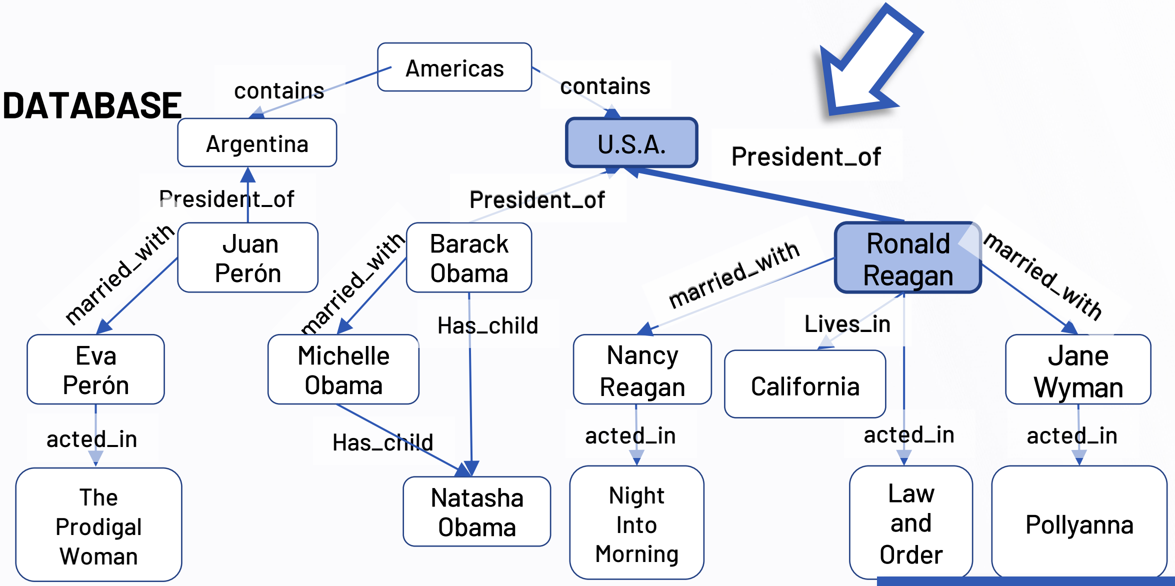


We show how to apply IR approaches
to graphs queries instead of keywords

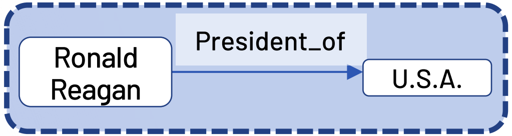
Bag Model for Graphs

Lissandrini et al. [WebConf '20]

DATABASE



The User Search



How can we exploit the document model?

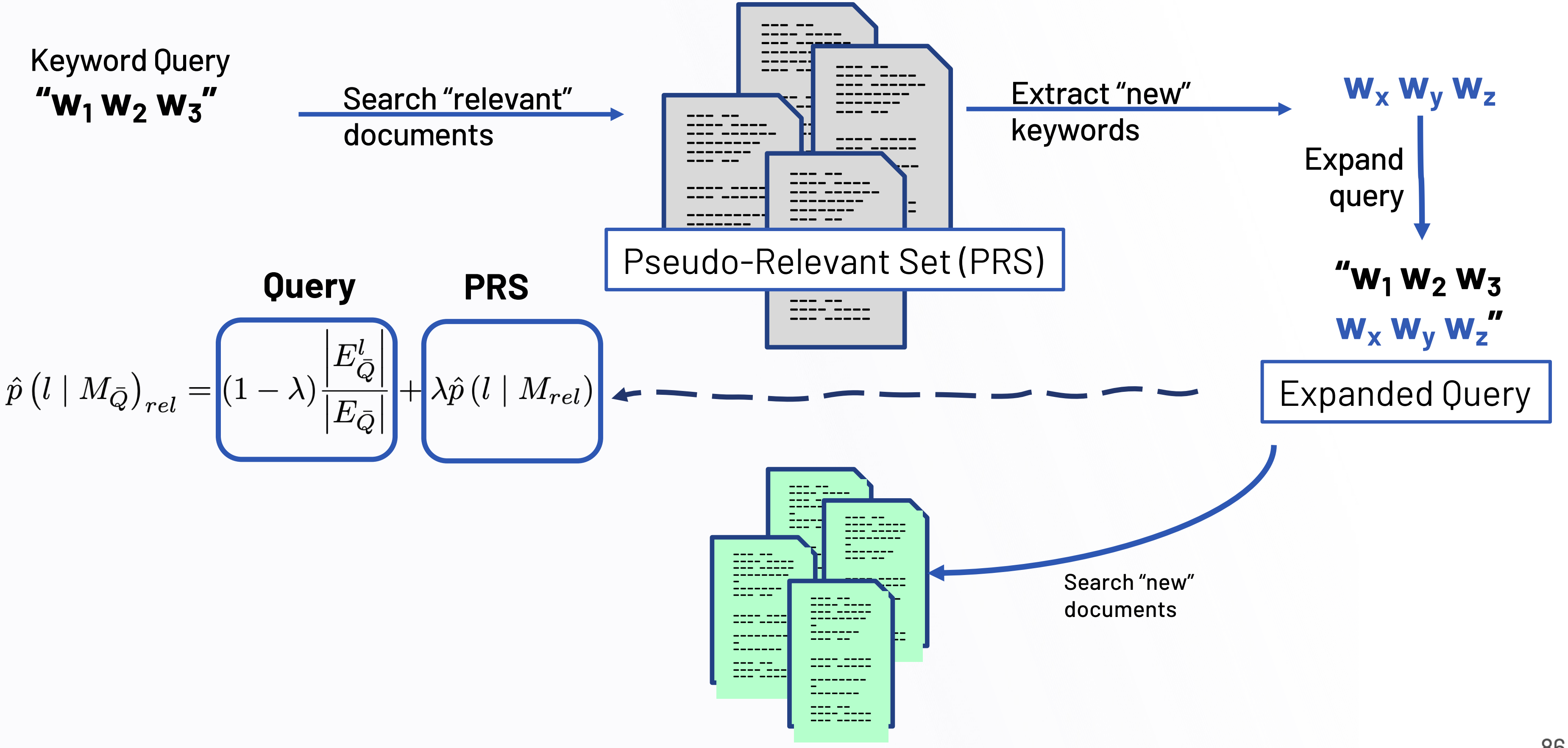
The Bag-of-Labels Model

{President_of, contains, married_with, married_with, acted_in, lives_in}

- Graphs can be modeled as **Bag of Words**
- Describes **MORE than what is in the query**

Pseudo Relevance Feedback for Document Search

Lissandrini et al. [WebConf '20]



$$\hat{p}\left(l \mid M_{\bar{Q}}\right)_{rel} = \overbrace{(1-\lambda)\frac{\left|E^{l}_{\bar{Q}}\right|}{\left|E_{\bar{Q}}\right|}}^{\text{Query}} + \overbrace{\lambda\hat{p}\left(l \mid M_{rel}\right)}^{\text{PRS}}$$

Pseudo Relevance Feedback Models

Lissandrini et al. [WebConf '20]

2 Models of Estimation
MLE & KL-Divergence

PRS

$$\hat{p}\left(l \mid M_{\bar{Q}}\right)_{rel} = (1-\lambda)\frac{\left|E^{l}_{\bar{Q}}\right|}{\left|E_{\bar{Q}}\right|} + \boxed{\lambda \hat{p}\left(l \mid M_{rel}\right)}$$

Maximum Likelihood Estimation

$$\hat{p}(l|M_{rel})_{MLE} \approx \sum_{\boxed{\bar{G}\in\bar{\mathcal{G}}_{rel}}} \overbrace{\hat{p}(l|M_{\bar{G}})\hat{p}(\bar{Q}|M_{\bar{G}})}^{\text{Frequent in the PRS}}$$

$$\hat{p}(\bar{Q}|M_{\bar{G}}) \propto \prod_{l\in\bar{Q}} \hat{p}(l|M_{\bar{G}})$$

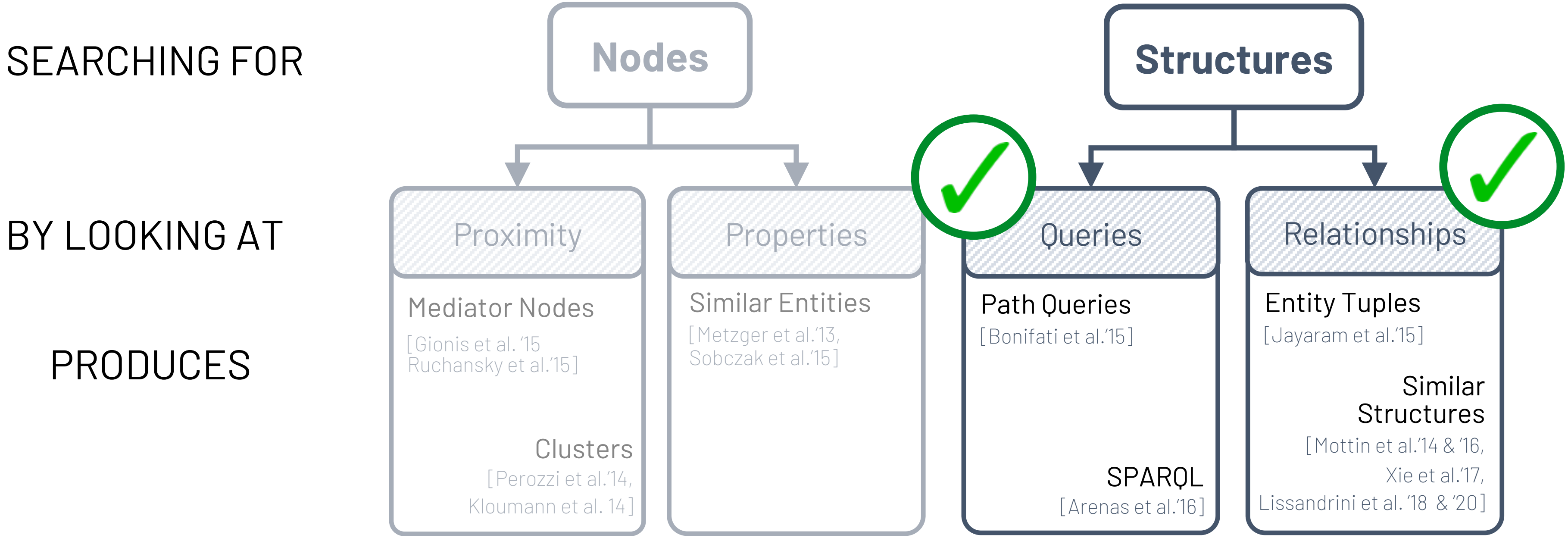
Exemplar Query Answers

KL-Divergence

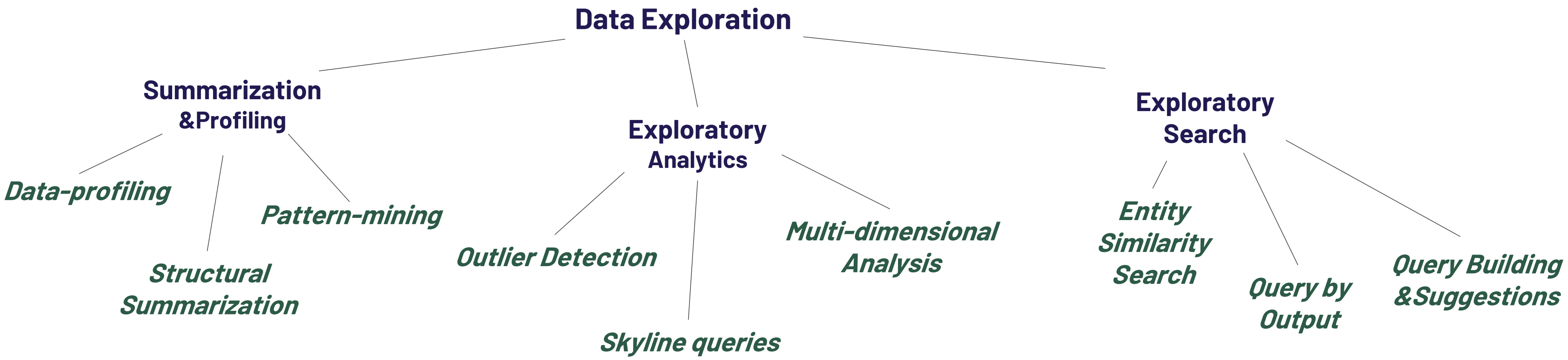
$$\hat{p}(l|M_{rel})_{KL} \propto$$

$$exp\left(\underbrace{\frac{1}{(1-\lambda)}\frac{1}{|\bar{\mathcal{G}}_{rel}|}\boxed{\sum_{\bar{G}}^{\bar{\mathcal{G}}_{rel}} \log\left(\hat{p}(l|M_{\bar{G}})\right)}}_{\text{Frequent in the PRS}} - \underbrace{\frac{\lambda}{(1-\lambda)}\log\left(\hat{p}(l|\mathcal{K})\right)}_{\text{Frequent in the Graph}}\right)$$

SIMILARITY for GRAPHS



Data Exploration Methods



No Interaction and Personalization
Requires: No Domain Knowledge
Output: High Level Overview

Medium-High Interactivity
Requires: High-level Information Need
Output: Overview of Specific Aspects

High Interactivity and Personalization
Requires: Detailed Sample or Query Intent
Output: Detailed Answers

Overview: Goals/Tasks/Operations/Challenges

Goals

- Understand Overall Structure & Contents
- Identify Relevant Relationships & Attributes
- Extract Relevant Data & Insights

Tasks

- Summarization & Profiling
- Exploratory Search
- Exploratory Analytics

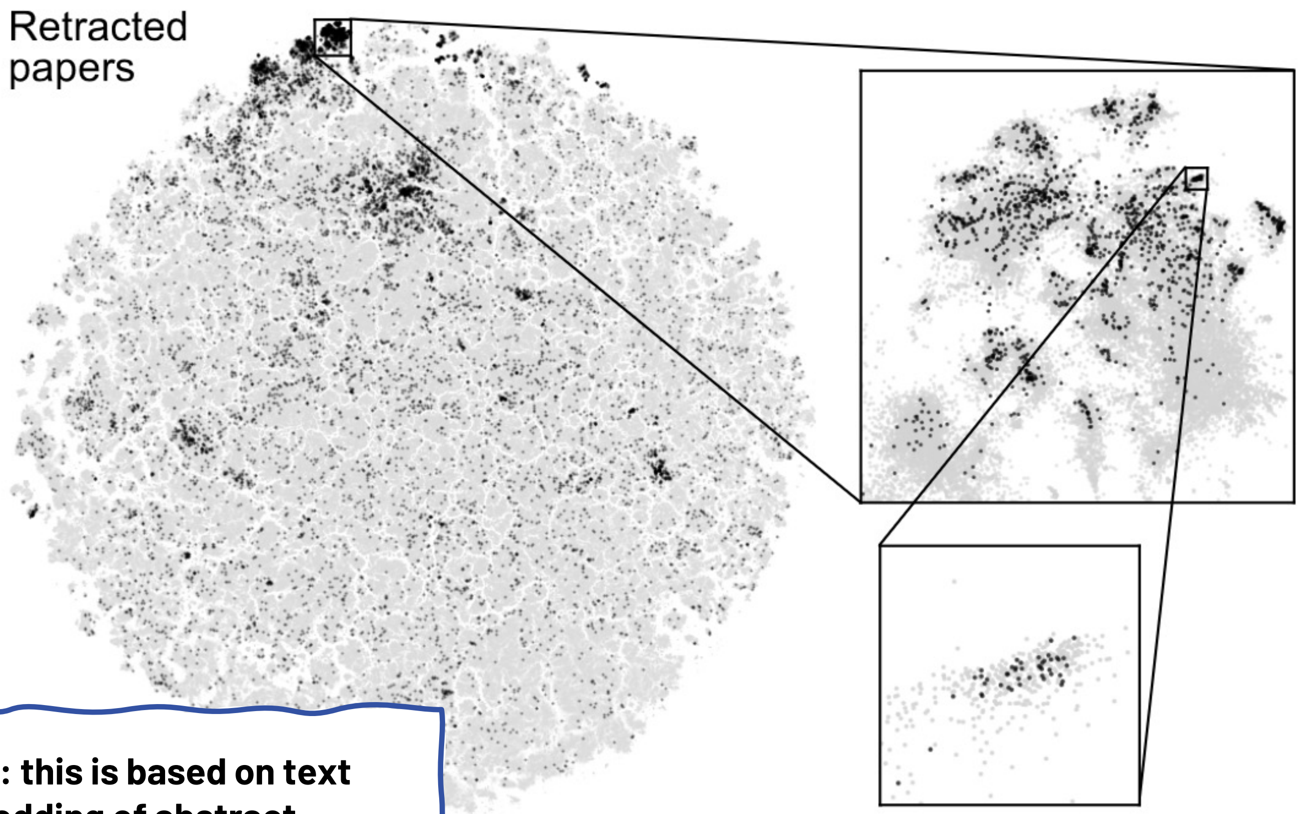
Operations

- Approximate Similarity
- Pattern Mining & Connectivity Search
- Progressive & Incremental Computation
- Iterative Query Reformulation
- Adaptive Storage & Indexing

Challenges

- Heterogeneity
- Evolution
- Vagueness
- Scale

A map of retracted papers (11k) in PubMed (21m).
There are clear clusters and we believe it's paper mill activity.



Note: this is based on text embedding of abstract

The landscape of biomedical research
Rita González-Márquez, Luca Schmidt, Benjamin M. Schmidt, Philipp Berens, Dmitry Kobak
doi: https://doi.org/10.1101/2023.04.10.536208

Explore a Paper Mill activity network

https://en.wikipedia.org/wiki/Research_paper_mill

You have access to a large citation graph with authors, papers, venues, affiliations, years, citations.

You want to analyze retracted papers and identify possible paper mill activities.

What model and methods do you apply?

Mining Patterns from a Query log

You have access to the log of all SPARQL queries submitted to a large KG DBMS.

What graph analysis approached can you apply to this data.

Define how you would approach it.

Can you find patterns? Communities?

What can a pattern or community tell you?

How can you use this information?

Acknowledgments

We would like to thank the authors of the papers who kindly provided us the slides

Angela Bonifati, Radu Ciucianu, Marcelo Arenas, Gonzalo Diaz, Egor Kostylev, Yaacov Weiss, Sarah Cohen, Fotis Psallidas, Li Hao, Chan Chee Yong, Ilaria Bordino, Mohamed Yakout, Kris Ganjam, Kaushik Chakrabati, Thibault Sellam, Rohit Singh, Maeda Hanafi, Dmitri Kalashnikov, Marcin Sydow, Mingzhu Zhu, Yoshiharu Ishikawa, Daniel Deutch, Nandish Jayaram, Paolo Papotti, Bryan Perozzi, Kiriaki Dimitriadou, Yifei Ma, Natali Ruchansky, Quoc Trung Tran, Hastagiri Prakash Vanchinathan

... and many others (see references)

References

M. Arenas, G. I. Diaz, and E. V. Kostylev. Reverse engineering sparql queries. WWW, 2016.

Agichtein, E. and Gravano, L. Snowball: Extracting relations from large plain-text collections. ICDL, 2000.

A.Bonifati, R.Ciucanu,and A.Lemay. Learning path queries on graph databases. EDBT, 2015.

A. Bonifati, R. Ciucanu, and S. Staworko. Learning join queries from user examples. TODS, 2016.

A. Bonifati, U. Comignani, E. Coquery, and R. Thion. Interactive mapping specification with exemplar tuples. SIGMOD, 2017.

I. Bordino, G. De Francisci Morales, I. Weber, and F. Bonchi. From machu picchu to rafting the urubamba river: anticipating information needs via the entity-query graph. WSDM, 2013.

D. Deutch and A. Gilad. Qplain: Query by explanation. ICDE, 2016.

References

G. Diaz, M. Arenas, and M. Benedikt. Sparqlbye: Querying rdf data by example. PVLDB, 2016.

K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In SIGMOD, 2014.

B. Eravci and H. Ferhatosmanoglu. Diversity based relevance feedback for time series search. PVLDB, 2013.

A. Gionis, M. Mathioudakis, and A. Ukkonen. Bump hunting in the dark: Local discrepancy maximization on graphs. ICDE, 2015.

M. F. Hanafi, A. Abouzied, L. Chiticariu, and Y. Li. Synthesizing extraction rules from user examples with seer. SIGMOD, 2017.

He, J., Veltri, E., Santoro, D., Li, G., Mecca, G., Papotti, P. and Tang, N. Interactive and deterministic data cleaning. SIGMOD, 2016.

Y. Ishikawa, R. Subramanya, and C. Faloutsos. Mindreader: Querying databases through multiple examples. VLDB, 1998.

References

N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri. Querying knowledge graphs by example entity tuples. TKDE, 2015.

H. Li, C.-Y. Chan, and D. Maier. Query from examples: An iterative, data-driven approach to query construction. PVLDB, 2015.

M. Lissandrini, D. Mottin, Y. Velegrakis, T. Palpanas. Multi-Example Search in Rich Information Graphs ICDE 2018

Y. Ma, T.-K. Huang, and J. G. Schneider. Active search and bandits on graphs using sigma-optimality. UAI, 2015.

S. Metzger, R. Schenkel, and M. Sydow. Qbees: query by entity examples. CIKM, 2013.

D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Searching with xq: the exemplar query search engine. SIGMOD, 2014.

D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Exemplar queries: a new way of searching. VLDB J., 2016.

B. Perozzi, L. Akoglu, P. Iglesias Sanchez, and E. Müller. Focused clustering and outlier detection in large attributed graphs. KDD, 2014.

References

F. Psallidas, B. Ding, K. Chakrabarti, and S. Chaudhuri. S4: Top-k spreadsheet-style search for query discovery. SIGMOD, 2015.

R. Rolim, G. Soares, L. D'Antoni, O. Polozov, S. Gulwani, R. Gheyi, R. Suzuki, and B. Hartmann. Learning syntactic program transformations from examples. ICSE, 2017.

N. Ruchansky, F. Bonchi, D. García-Soriano, F. Gullo, and N. Kourtellis. The minimum wiener connector problem. SIGMOD, 2015.

T. Sellam and M. Kersten. Cluster-driven navigation of the query space. TKDE, 2016.

Y. Shen, K. Chakrabarti, S. Chaudhuri, B. Ding, and L. Novik. Discovering queries based on example tuples. SIGMOD, 2014.

R. Singh. Blinkfill: Semi-supervised programming by example for syntactic string transformations. PVLDB, 2016.

G. Sobczak, M. Chochół, R. Schenkel, and M. Sydow. iqbees: Towards interactive semantic entity search based on maximal aspects. Foundations of Intelligent Systems, 2015.

References

Y. Su, S. Yang, H. Sun, M. Srivatsa, S. Kase, M. Vanni, and X. Yan. Exploiting relevance feedback in knowledge graph search. KDD, 2015.

Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. Query reverse engineering. VLDB J., 2014.

H. P. Vanchinathan, A. Marfurt, C.-A. Robelin, D. Koss- mann, and A. Krause. Discovering valuable items from massive data. In KDD, 2015.

C.Wang, A.Cheung, and R.Bodik. Interactive query synthesis from input-output examples. In SIGMOD, 2017.

C. Wang, A. Cheung, and R. Bodik. Synthesizing highly expressive sql queries from input-output examples. In PLDI, 2017.

Y. Y. Weiss and S. Cohen. Reverse engineering spj-queries from examples. SIGMOD, 2017.

M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: Entity augmentation and attribute discovery by holistic matching with web tables. SIGMOD, 2012.

M. Zhu and Y.-F. B. Wu. Search by multiple examples. WSDM, 2014.

M. M. Zloof. Query by example. AFIPS NCC, 1975.

Further References

Davis Shurbert, An Introduction to Graph Homomorphisms
http://buzzard.ups.edu/courses/2013spring/projects/davis-homomorphism-ups-434-2013.pdf

Andreas Schmidt, Iztok Savnik, Conference on Advances in Databases, Knowledge, and Data Applications, Overview of Regular Path Queries in Graphs
https://www.iaria.org/conferences2015/filesDBKDA15/graphsm_overview_of_regular_path_queries_in_graphs.pdf

Jure Leskovec, CS224W: Machine Learning with Graphs | 2019 |
Lecture 3-Motifs and Structural roles in networks
http://snap.stanford.edu/class/cs224w-2019/slides/03-motifs.pdf

Davide Mottin and Emmanuel Müller, Graph Exploration:
Let me Show what is relevant in your graph, KDD tutorial
https://mott.in/slides/KDD2018-Tutorial-Compressed.pdf

Kolaczyk, E.D., Statistical Analysis of Network Data, Chapter 5: Sampling and Estimation in Network Graphs.
https://link.springer.com/chapter/10.1007/978-0-387-88146-1_5

Jure Leskovec, Christos Faloutsos, Sampling from Large Graphs
https://dl.acm.org/doi/pdf/10.1145/1150402.1150479