Machine Learning course

Part 4a: Learning from Heterogeneous Data

26 January 2021

Nada Lavrač

Jožef Stefan Institute, University of Ljubljana

Ljubljana, Slovenia

Learning from heterogeneous data

- Motivation for heterogeneous data analysis
- Semantic relational learning
  - Propositionalization approach (repeated from Lesson 3)
  - Top-down search for rules with Hedwig
  - Reducing the search with NetSDM
- Propositionalization of heterogeneous information networks
  - TEHmINE
  - HINMINE
- Practical exercises with HINMINE

Motivation for heterogeneous data analysis: Various data types

- **Relational data**
  - Single relation → Machine learning from tabular data
  - Multiple relations → Relational learning and ILP from multiple tables: one target data table and background knowledge encoded in related data tables

    (recall relational learning from Lesson 3)

- **Text data**

  → Text mining and natural language processing

  (recall text mining in wordification from Lesson 3)

Motivation for heterogeneous data analysis: Various data types

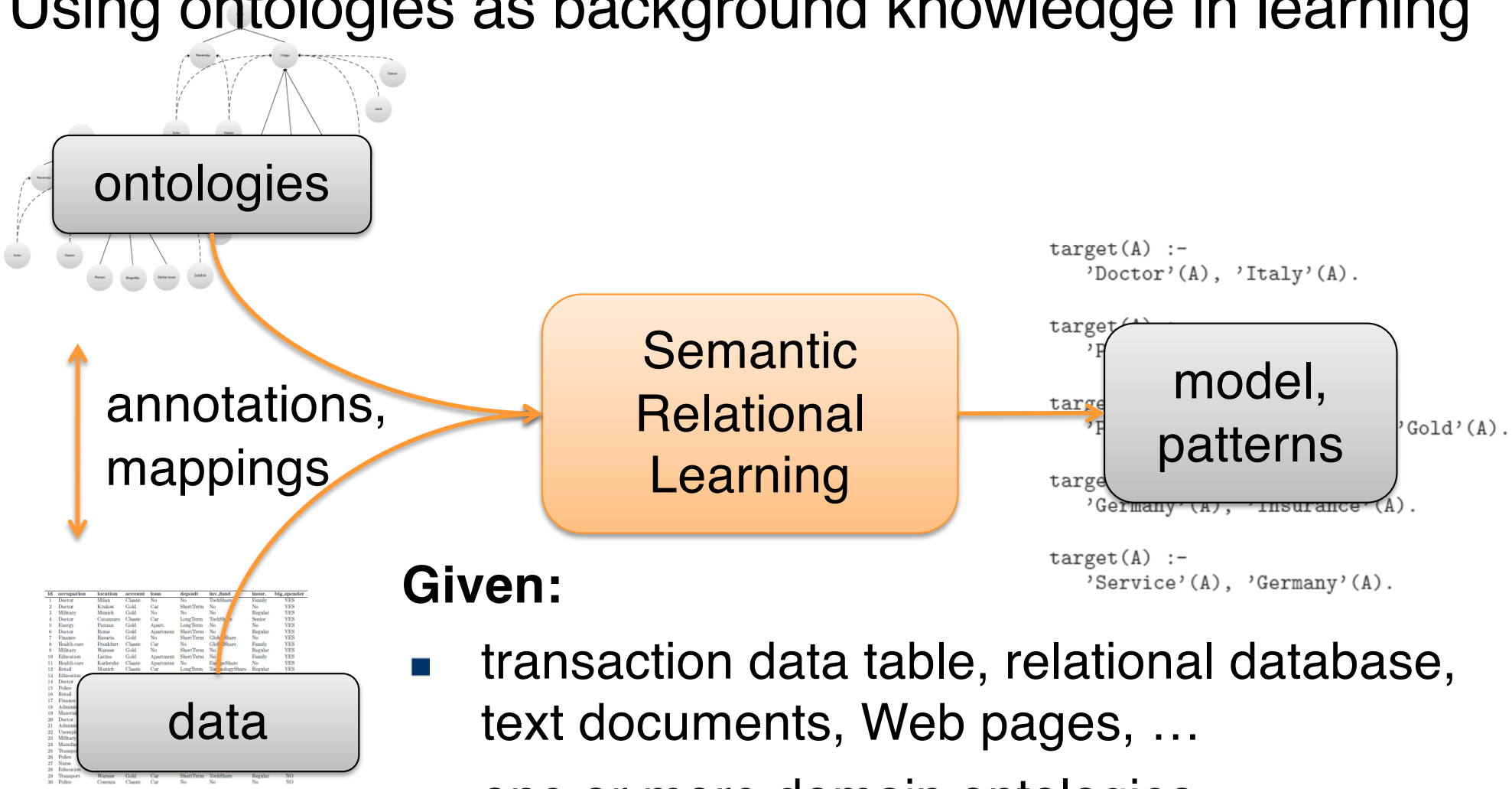
- **Heterogeneous data**
  - Different data types: entities, tables, texts, pictures, …
  - Involves interconnected entities

  → Semantic relational learning – data analysis with background knowledge in the form of ontologies, (hierarchical relations between entities/concepts)
  (this lesson, including Hedwig and NetSDM)

  → Graph and heterogeneous information network analysis
  (this lesson, including TEHmINE and HINMINE)

Semantic Relational Learning (SDM)

Using ontologies as background knowledge in learning



Given:

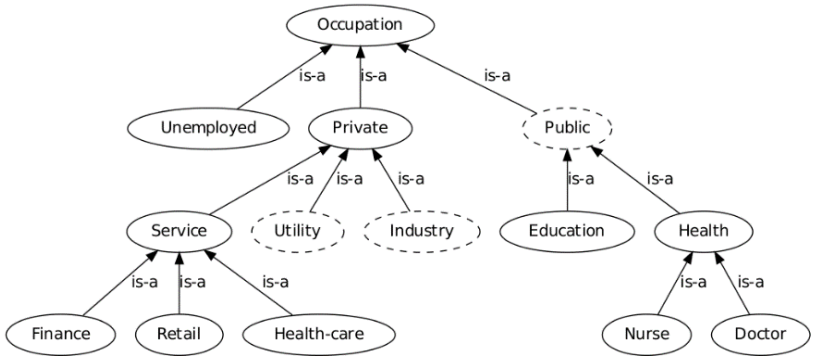
- transaction data table, relational database, text documents, Web pages, …
- one or more domain ontologies

Find: a classification model, a set of patterns

Motivational example

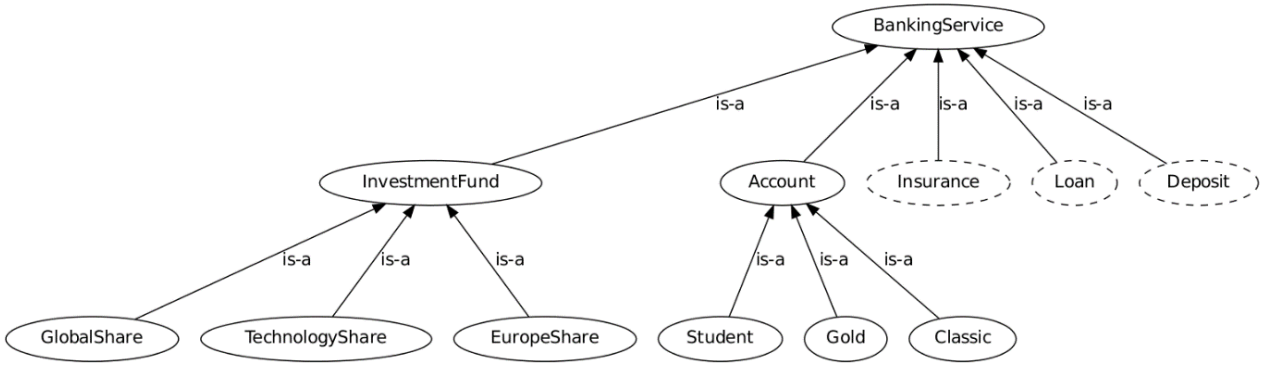
client

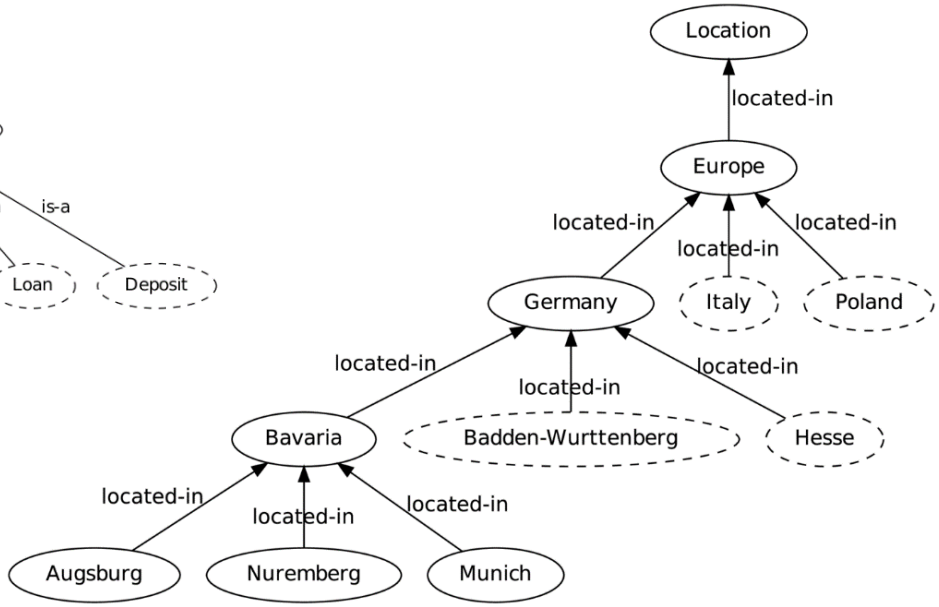
id	occupation	location	account	...	big spender
0	Doctor	Munich	Gold	...	yes
1	Nurse	Rome	Classic	...	yes
2	Finance	Krakow	Gold	...	yes
...	...	...	...		
27	Retail	Bologna	Classic	...	no
28	Finance	Nuremberg	Classic	...	no
29	Nurse	Augsburg	Student	...	no



married

client1Id	client2Id
0	11
1	2





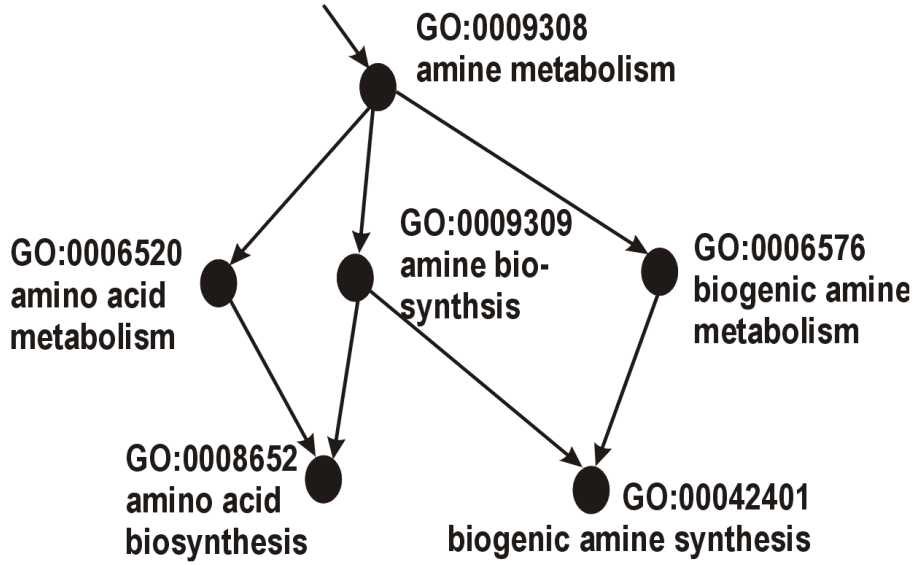
Motivational example

```
big_spender(X) ← married(X, Y),
                 has_occupation(Y, healthSector),
                 uses_service(Y, goldAcc)

big_spender(X) ← has_occupation(X, doctor),
                 uses_service(X, deposit)

big_spender(X) ← lives_in(X, germany),
                 has_occupation(X, serviceSector),
                 uses_service(X, investment_fund)
```

Example biomedical ontology GO

Using domain ontologies as background knowledge, e.g., using the **Gene Ontology** (GO)

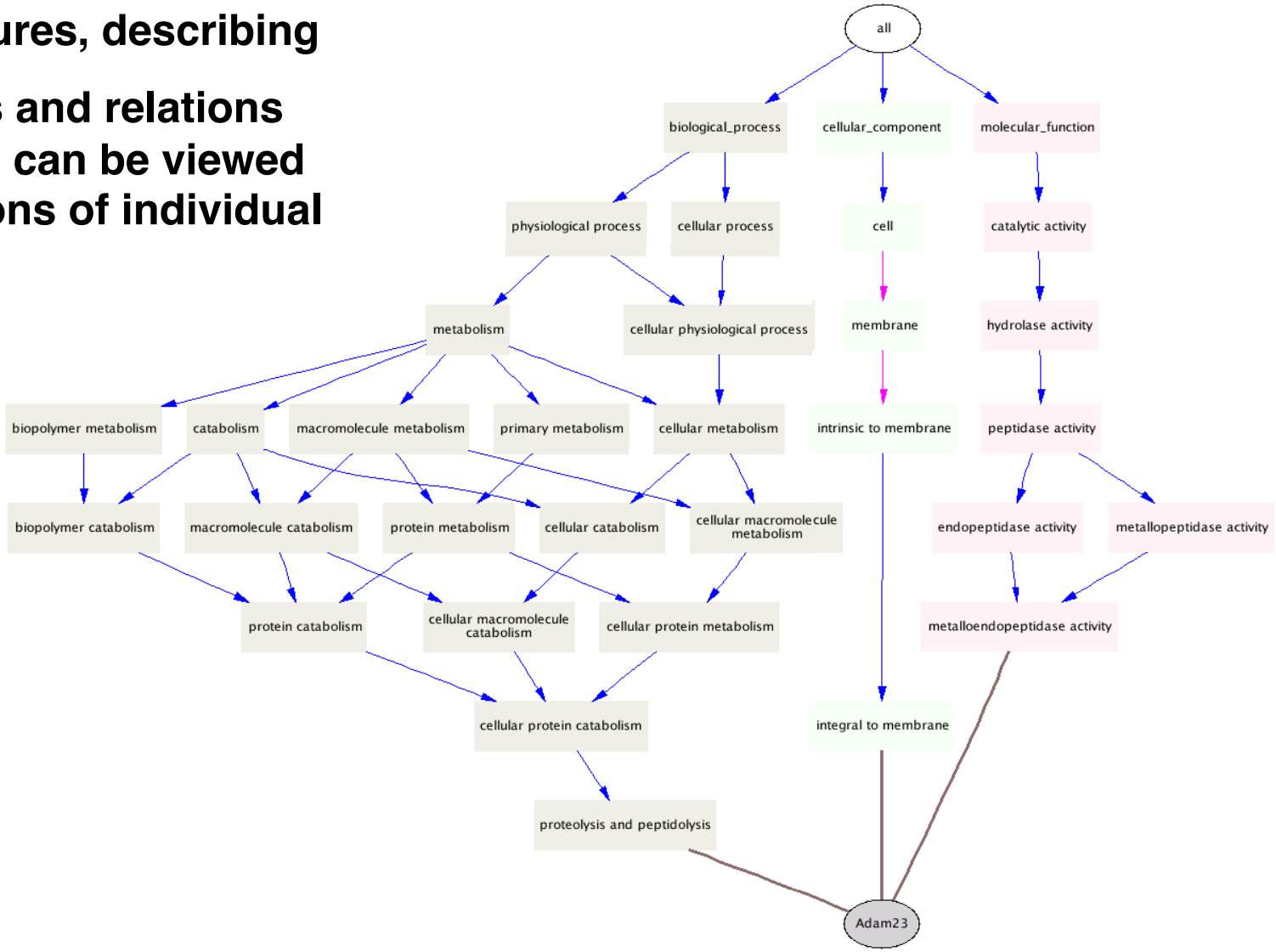
- GO is a database of terms,
  describing gene sets in terms of their
  - functions (over 12,000)
  - processes (over 2,000)
  - components (over 7,500)
- Genes are annotated
  to GO terms
- Terms are connected
  (is_a, part_of)
- Levels represent terms generality



Using GO as background knowledge e.g., in DNA microarray data analysis

First-order features, describing

gene properties and relations between genes, can be viewed as generalisations of individual genes



Gene ontology encoded in Prolog: Example DNA microarray data analysis

- Ontology terms and relations encoded as logical facts in Prolog, e.g.

```
component(gene2532,'GO:0016020').
function(gene2534,'GO:0030554').
process(gene2534,'GO:0007243').
interaction(gene2534,gene4803).
```

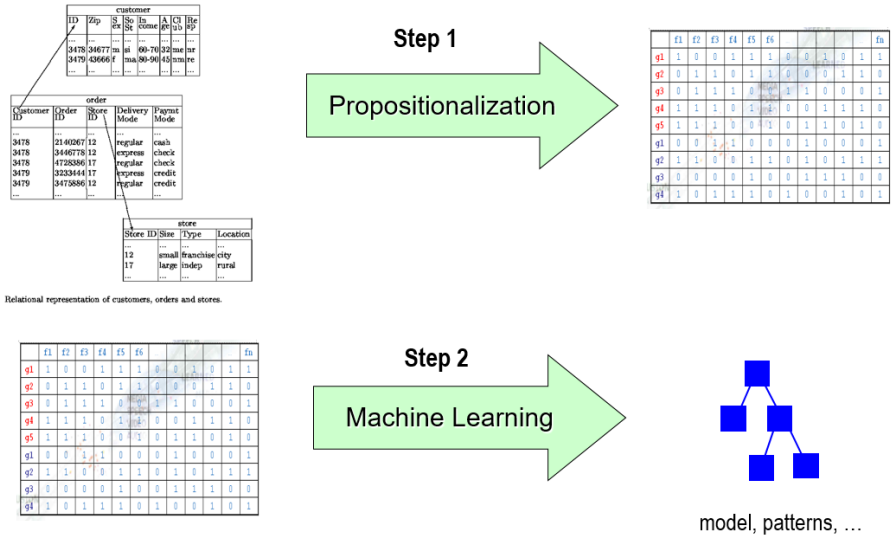
- Gene labels also encoded as facts, e.g. positive and negative examples

```
diffexp(gene64499).
diffexp(gene2534).
diffexp(gene5199).
diffexp(gene1052).
diffexp(gene6036).
…
```

```
random(gene7443).
random(gene9221).
random(gene2339).
random(gene9657).
random(gene19679).
…
```

RSD: Propositionalization approach to Semantic Relational learning

- Recall RSD from Lesson 3
- Input
  - Input data are Prolog facts,
  - Background knowledge in the form of ontologies is encoded as Prolog facts or rules
- Propositionalization with RSD
  - Construct relational features
  - Determine truth values of features
  - Learn rules with CN2-SD



RSD: Propositionalization approach to Semantic Relational learning

Take ontology terms represented as logical facts in Prolog, e.g.

```
component(gene2532,'GO:0016020').
function(gene2534,'GO:0030554').
process(gene2534,'GO:0007243').
interaction(gene2534,gene4803).
```

1. Automatically generate generalized relational features:

```
f(2,A):-component(A,'GO:0016020').
f(7,A):-function(A,'GO:0030554').
f(11,A):-process(A,'GO:0007243').
f(224,A):- interaction(A,B), function(B,'GO:0016787'),
           component(B,'GO:0043231').
```

2. Propositionalization: Determine truth values of features

3. Learn rules by a subgroup discovery algorithm CN2-SD

Step 1: RSD feature construction

Construction of first order features, with support > *min_support*

```
f(7,A):-function(A,'GO:0046872').
f(8,A):-function(A,'GO:0004871').
f(11,A):-process(A,'GO:0007165').
f(14,A):-process(A,'GO:0044267').
f(15,A):-process(A,'GO:0050874').
f(20,A):-function(A,'GO:0004871'), process(A,'GO:0050874').
f(26,A):-component(A,'GO:0016021').
f(29,A):- function(A,'GO:0046872'), component(A,'GO:0016020').
f(122,A):-interaction(A,B),function(B,'GO:0004872').
f(223,A):-interaction(A,B),function(B,'GO:0004871'),
    process(B,'GO:0009613').
f(224,A):-interaction(A,B),function(B,'GO:0016787'),
    component(B,'GO:0043231').
```

existential

Step 2: RSD Propositionalization: Example DNA microarray data analysis

```
diffexp(gene64499).
diffexp(gene2534).
diffexp(gene5199).
diffexp(gene1052).
diffexp(gene6036).
```

```
random(gene7443).
random(gene9221).
random(gene2339).
random(gene9657).
```

	f1	f2	f3	f4	f5	f6	…				…	fn
g1	1	0	0	1	1	1	0	0	1	0	1	1
g2	0	1	1	0	1	1	0	0	0	1	1	0
g3	0	1	1	1	0	0	1	1	0	0	0	1
g4	1	1	1	0	1	1	0	0	1	1	1	0
g5	1	1	1	0	0	1	0	1	1	0	1	0
g100	0	0	1	1	0	0	0	1	0	0	0	1
g101	1	1	0	0	1	1	0	1	0	1	1	1
g102	0	0	0	0	1	0	0	1	1	1	0	0
g103	1	0	1	1	1	0	1	0	0	1	0	1

Step 3: RSD rule construction with CN2-SD

	f1	f2	f3	f4	f5	f6	…				…	fn
g1	1	0	0	1	1	1	0	0	1	0	1	1
g2	0	1	1	0	1	1	0	0	0	1	1	0
g3	0	1	1	1	0	0	1	1	0	0	0	1
g4	1	1	1	0	1	1	0	0	1	1	1	0
g5	1	1	1	0	0	1	0	1	1	0	1	0
g100	0	0	1	1	0	0	0	1	0	0	0	1
g101	1	1	0	0	1	1	0	1	0	1	1	1
g102	0	0	0	0	1	0	0	1	1	1	0	0
g103	1	0	1	1	1	0	1	0	0	1	0	1

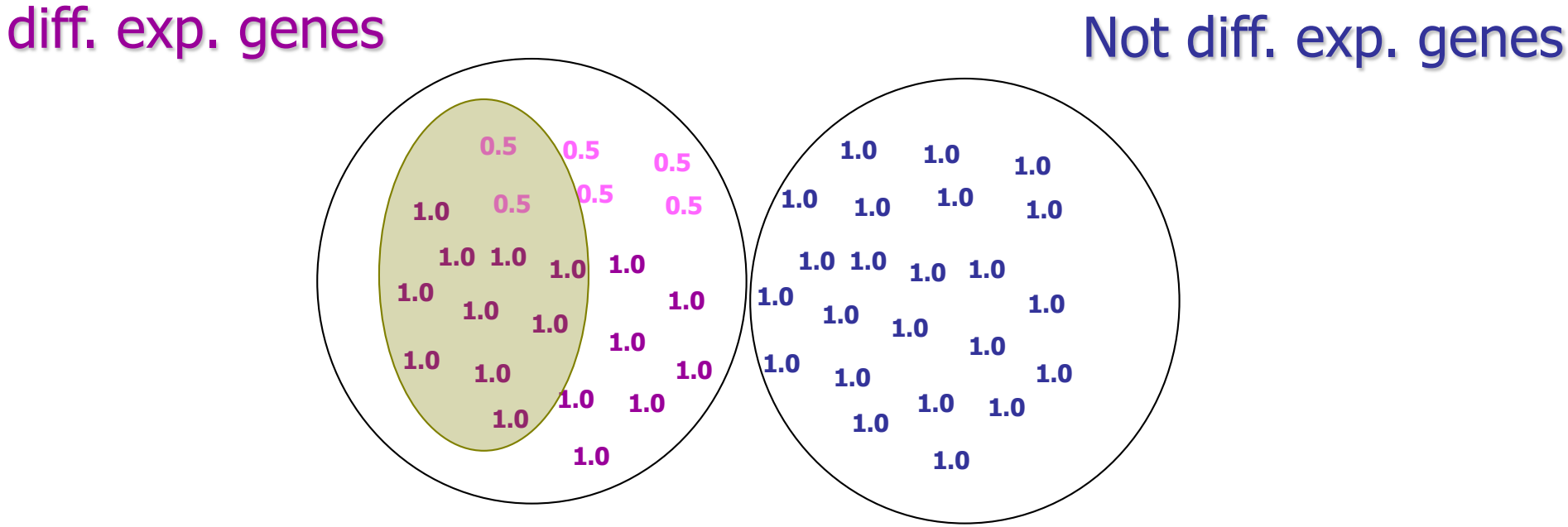
differentially expressed

IF

f2 and f3

[4,0]

Subgroup Discovery with CN2-SD: Weighted covering approach



RSD naturally uses gene weights in its procedure for repetitive subgroup generation, via its heuristic rule evaluation: weighted relative accuracy

Summary: Semantic relational learning with RSD in two main steps

- **Step 1:** Construct relational logic features of genes such as

  interaction(g, G) & function(G, protein_binding)
  (*g interacts with another gene whose functions include protein binding*)

  and propositional table construction with features as attributes

- **Step 2:** Use these features to discover and describe subgroups of genes that are differentially expressed in contrast with RANDOM genes (randomly selected genes with low differential expression).

- Sample subgroup description:

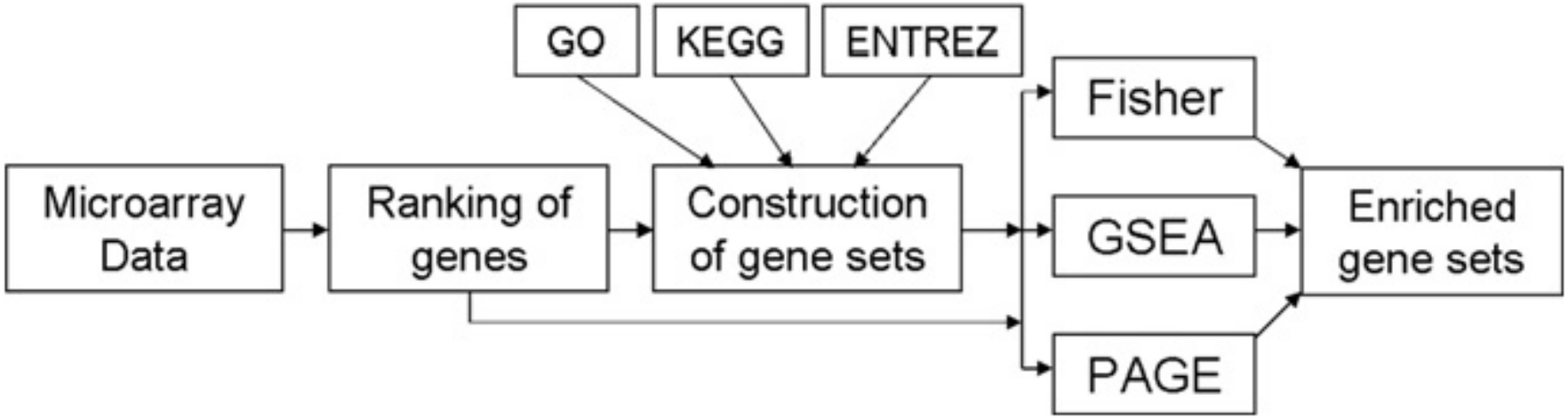
```
diffexp(A) :- interaction(A,B) AND
              function(B,'GO:0004871') AND
              process(B,'GO:0009613')
```

Semantic Data Mining in Orange4WS

- Slides 19-25 are supplementary to this lecture, for illustrative purposes only
- Illustrating a special purpose Semantic Data Mining algorithm SEGS
  - discovers interesting gene group descriptions as conjunctions of ontology concepts from GO, KEGG and Entrez
  - integrates public gene annotation data through relational features
  - SEGS algorithm (Trajkovski, Železny, Lavrač and Tolar, JBI 2008) is available in Orange4WS

Semantic subgroup discovery with SEGS

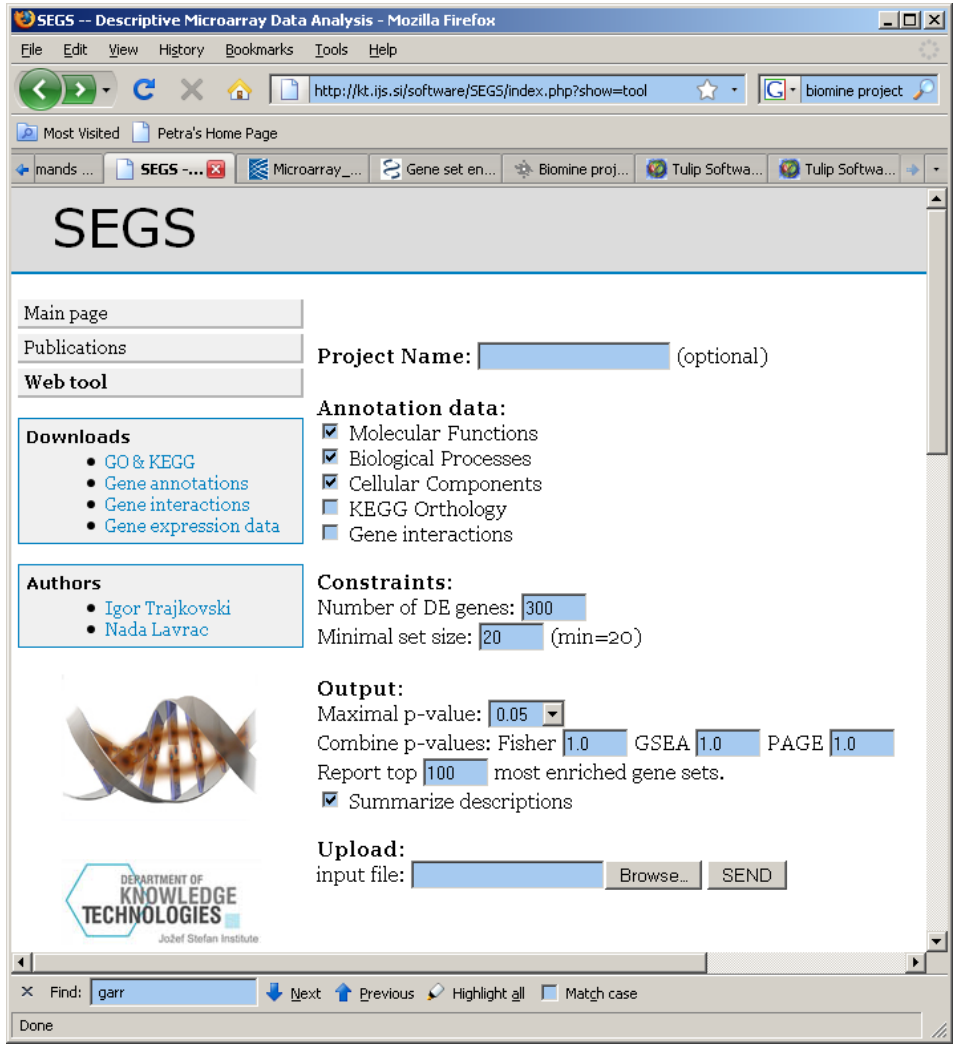
- SEGS workflow is implemented in the Orange4WS data mining environment

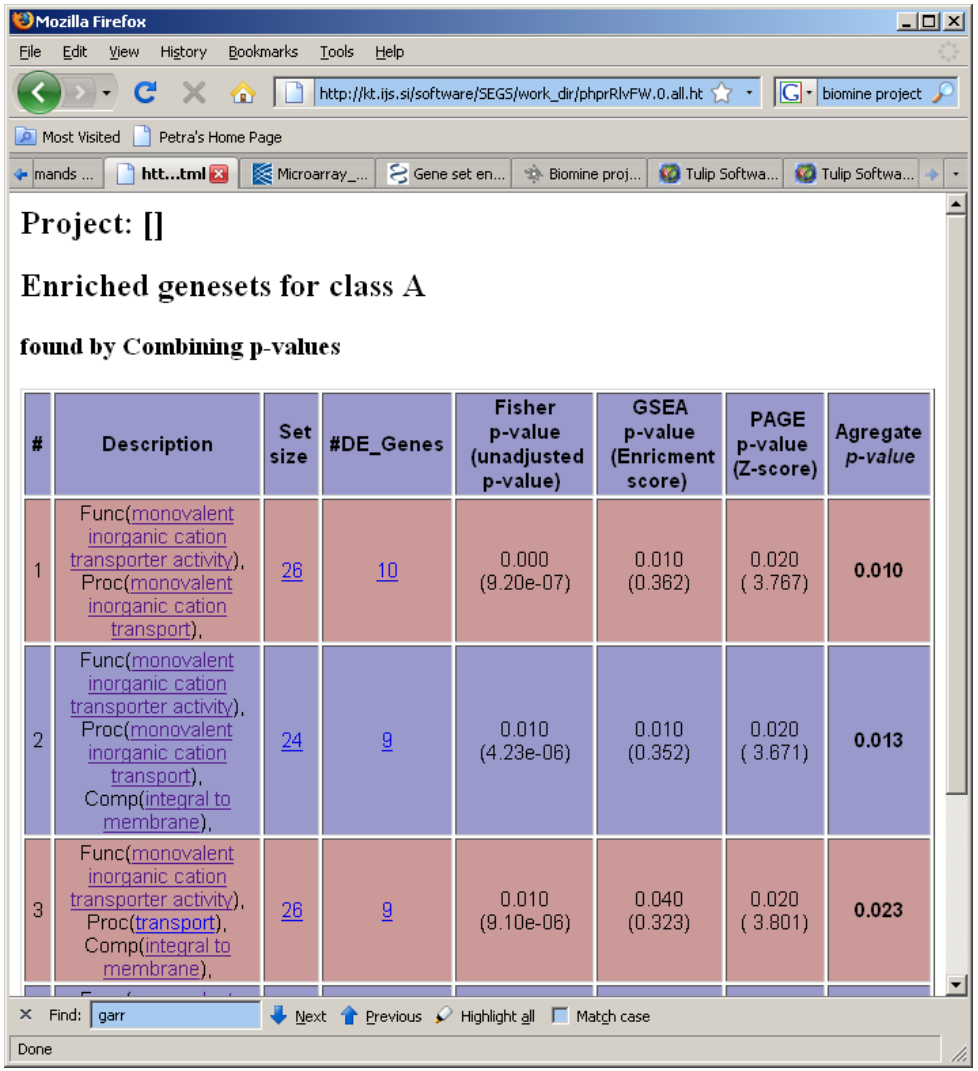


- SEGS is also implemented also as a Web applications

(Trajkovski et al., IEEE TSMC 2008, Trajkovski et al., JBI 2008)

Semantic subgroup discovery with SEGS



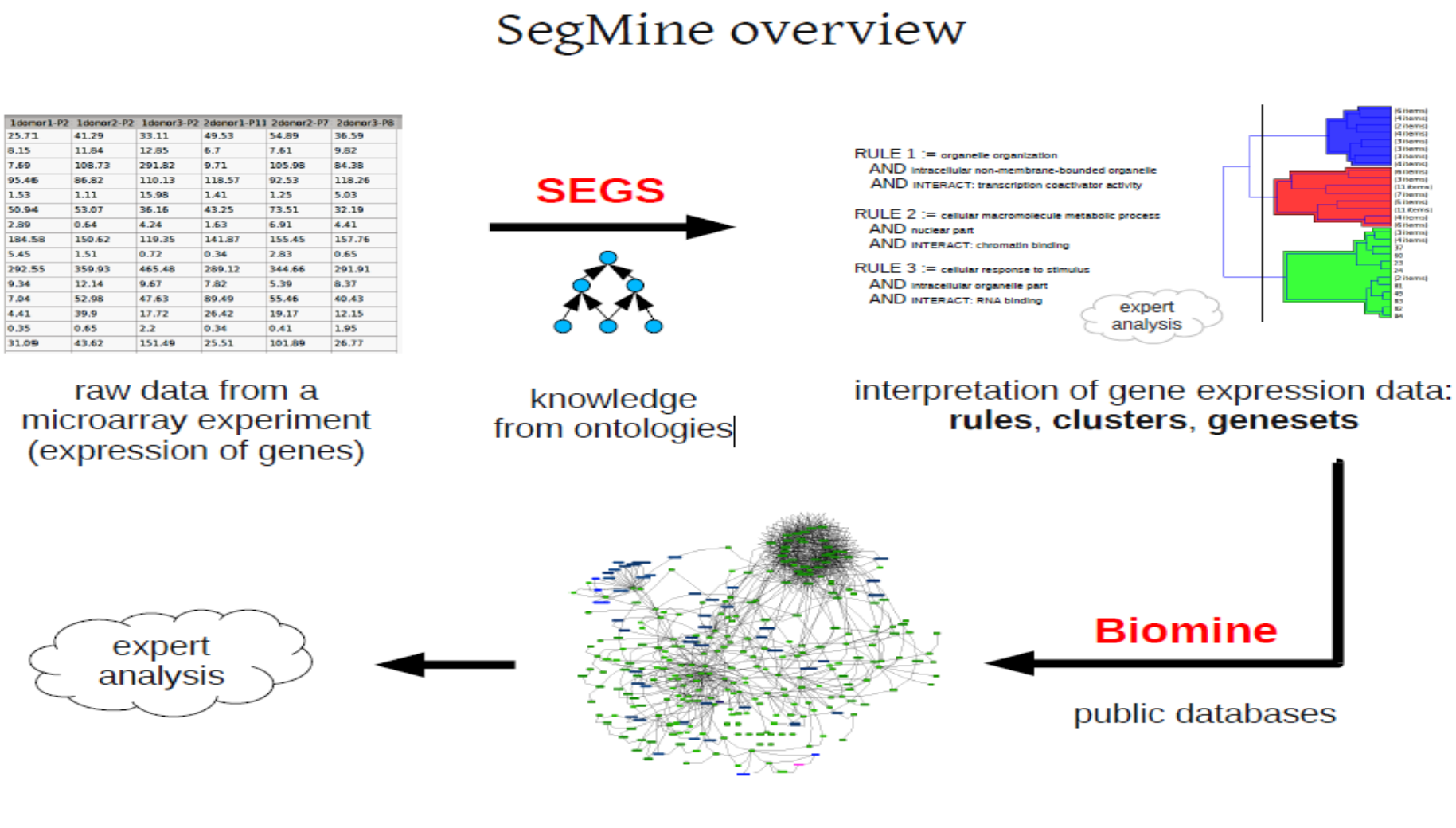


#	Description	Set size	#DE_Genes	Fisher p-value (unadjusted p-value)	GSEA p-value (Enricment score)	PAGE p-value (Z-score)	Agregate *p-value*
1	Func(monovalent inorganic cation transporter activity), Proc(monovalent inorganic cation transport),	26	10	0.000 (9.20e-07)	0.010 (0.362)	0.020 (3.767)	**0.010**
2	Func(monovalent inorganic cation transporter activity), Proc(monovalent inorganic cation transport), Comp(integral to membrane),	24	9	0.010 (4.23e-06)	0.010 (0.352)	0.020 (3.671)	**0.013**
3	Func(monovalent inorganic cation transporter activity), Proc(transport), Comp(integral to membrane),	26	9	0.010 (9.10e-06)	0.040 (0.323)	0.020 (3.801)	**0.023**

BioMine knowledge graph exploration engine (Toivonnen et al.)

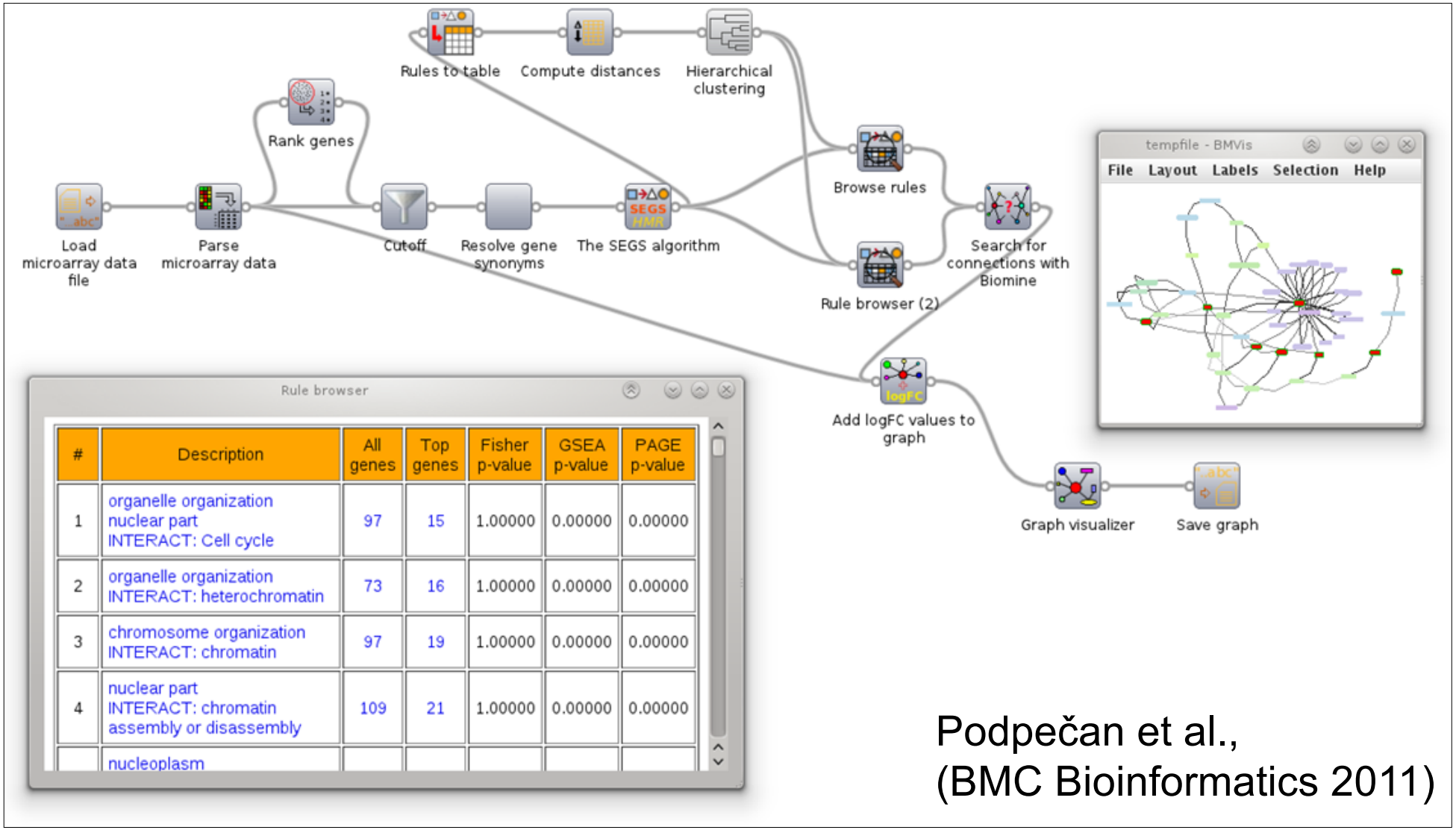
- SEGS has been combined with link discovery using BioMine (Toivonen et al.) in the SegMine workflow
- **BioMine graph** contains information from public databases, including annotated sequences, proteins, orthology groups, genes and gene expressions, gene and protein interactions, PubMed articles, and different ontologies.
  - **nodes (~1 mio)** correspond to different concepts (such as gene, protein, domain, phenotype, biological process, tissue)
  - **semantically labeled edges (~7 mio)** connect related concepts
- **BioMine query engine** answers queries to potentially discover new links between entities by sophisticated graph exploration algorithms

SegMine: Complex SDM methodology



(Podpečan et al., BMC Bioinformatics 2011)

SegMine implementation in Orange4WS platform

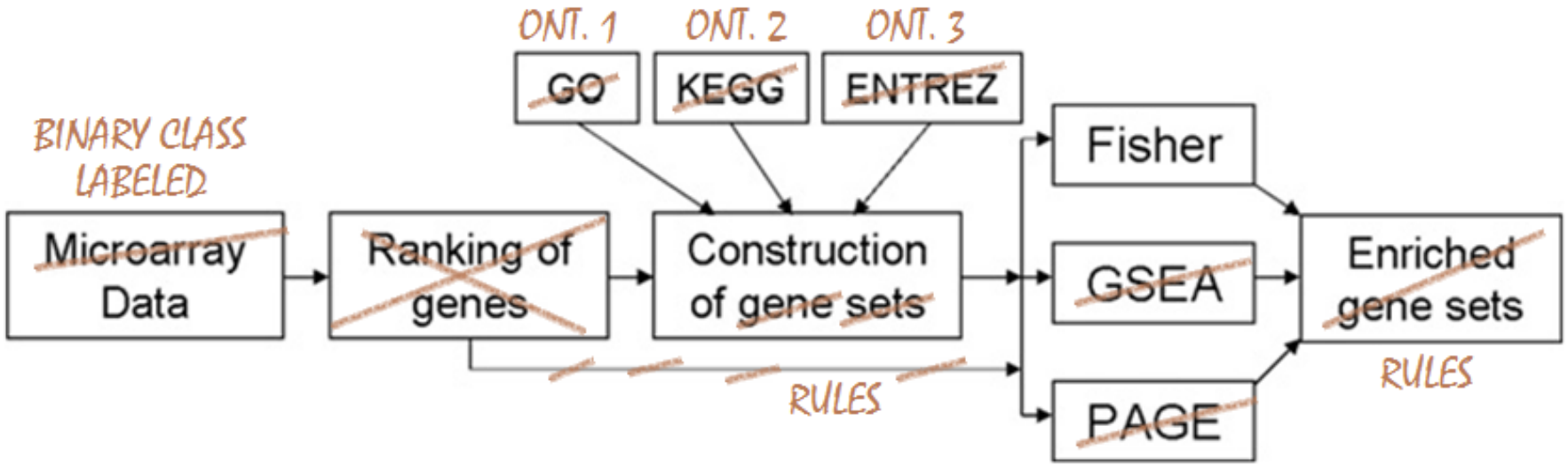


#	Description	All genes	Top genes	Fisher p-value	GSEA p-value	PAGE p-value
1	organelle organization nuclear part INTERACT: Cell cycle	97	15	1.00000	0.00000	0.00000
2	organelle organization INTERACT: heterochromatin	73	16	1.00000	0.00000	0.00000
3	chromosome organization INTERACT: chromatin	97	19	1.00000	0.00000	0.00000
4	nuclear part INTERACT: chromatin assembly or disassembly	109	21	1.00000	0.00000	0.00000
	nucleoplasm					

Podpečan et al.,
(BMC Bioinformatics 2011)

SDM-SEGS: Generalizing SEGS

- SDM-SEGS: a semantic data mining system generalizing SEGS



- Discovers subgroups both for ranked and labeled data
- Adapted to use any ontology in OWL format
- Implemented as a web service in Orange4WS or Taverna
- Implemented also as a workflow in ClowdFlows

SDM-Aleph: Generalizing Aleph

- An SDM system implemented using the popular ILP system Aleph [1]
- Adapted to accept ontologies in OWL
- Implemented as a WS in Orange4WS
- Implemented also as a workflow in ClowdFlows
- Same inputs/outputs as SDM-SEGS
- Any number of additional binary relations

[1] Ashwin Srinivasan
http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph.html

Hedwig general purpose Sematic relational learning algorithm

- Semantic Subgroup Discovery approach Hedwig
  - Speed from SDM-SEGS, due to exploiting the hierarchical structure in rule construction
  - Expressiveness from SDM-Aleph, allowing for any additional relations, and any # of ontologies
- Training examples and background knowledge in RDF
- Rule search space is structured via specialization predicates (e.g., subClassOf or user defined)
- Top down beam-search

  WRAcc, Lift, etc. as heuristics, Redundancy pruning

Hedwig rule construction by top-down search of the refinement graph

Empty rule: $y(X) \leftarrow$

Current rule: $y(X) \leftarrow p(X)$

Current rule specialization:

Replace predicate of a rule with a predicate that is a specialization of it

$y(X) \leftarrow q(X)$

Append a new predicate (next non-ancestor of p)

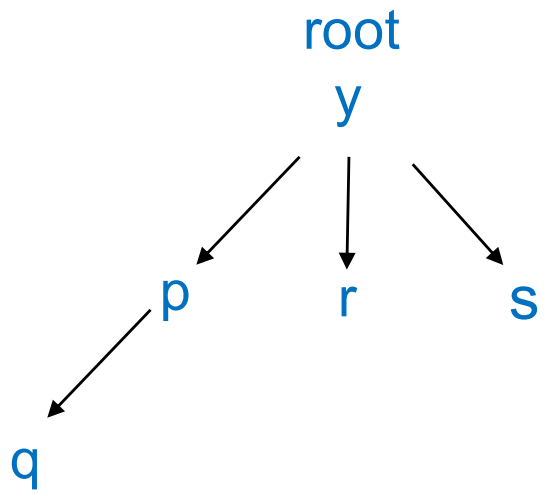
$y(X) \leftarrow p(X), r(X)$

Append a new binary predicate

$y(X) \leftarrow p(X), t(X, Y)$

(Negate a predicate)

$y(X) \leftarrow \neg p(X)$



Hedwig rule construction

Additional rule construction details

- Minimum support criterion
- Several rule scores: (WRAcc, lift, chisq, etc)
- Redundancy pruning (Hämäläinen, 2010)
- Significance: Fisher's exact test
- Multiple-hypothesis testing problem:
  - FWER: Holm-Bonferroni
  - FDR: Benjamini-Hochberg-Yekutieli

Hedwig

Algorithm 4.1: Hedwig's `induce`(E, B, c, k, α) procedure.

Input : Input examples E, background knowledge B, target class value c, beam size k, p-value threshold α

Output: Set of rules

```
rules ← [default_rule(E, c, B)]
while improvement(rules) do
    // Add specializations of each rule to the beam
    for rule ∈ rules do
        extend(rules, specialize(rule, B))
    end
    rules ← best(rules, k) // Select the top k rules
end
rules ← validate(rules, α) // Significance testing
return rules
```

Semantic relational learning: Related work

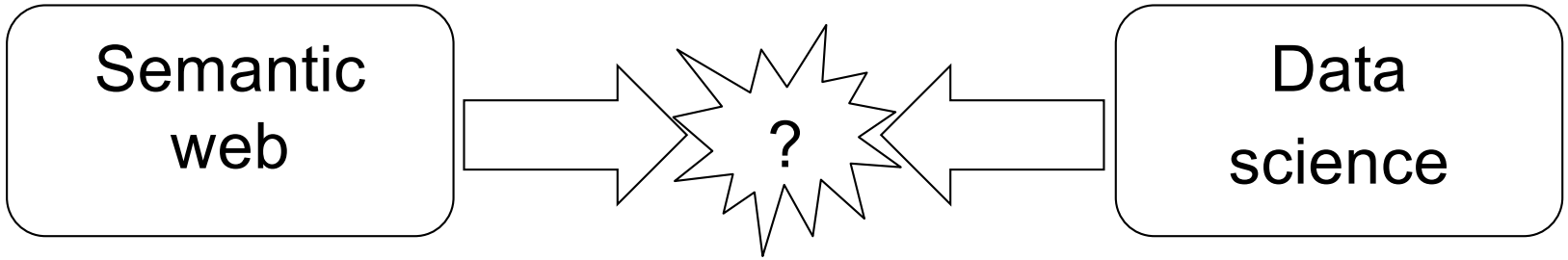
- SEGS - Gene set enrichment (Trajkovski et al, 2008)
- Taxonomies in propositionalization (Žakova and Železný, 2007)
- Association rules with taxonomies (Srikant and Agrawal, '95; Ayres and Santos, 2012; Manda et al, 2012)
- Feature selection in hierarchies (Garriga et al, 2008; Ristoski and Paulheim, 2014)
- DM ontology for meta-learning (Hilario et al, 2011)
- Description Logic learners (Kietz, 2002; Lehmann and Haase, 2009; Lawrynowicz 2011; Lisi, 2004-2009)

Learning from heterogeneous data

- Motivation for heterogeneous data analysis
- Semantic relational learning
  - Propositionalization approach (repeated from Lesson 3)
  - Top-down search for rules with Hedwig
  - Reducing the search with NetSDM
- Propositionalization of heterogeneous information networks
  - TEHmMINE
  - HINMINE
- Practical exercises

Advances in network analysis for SDM

The challenge is to fill the current gap between semantic web and data science: Which part of the semantic web is most important to my current interests?



Semantic Data Mining

+ Finds complex rules
+ Higly informative
- Computationally demanding
- Complexity grows exponentially

Fast
Scalable
Informative

Network analysis

+ Can process massive data
+ Fast, easy to calculate
- Less informative results

Challenge addressed in NetSDM

New challenge and methodology

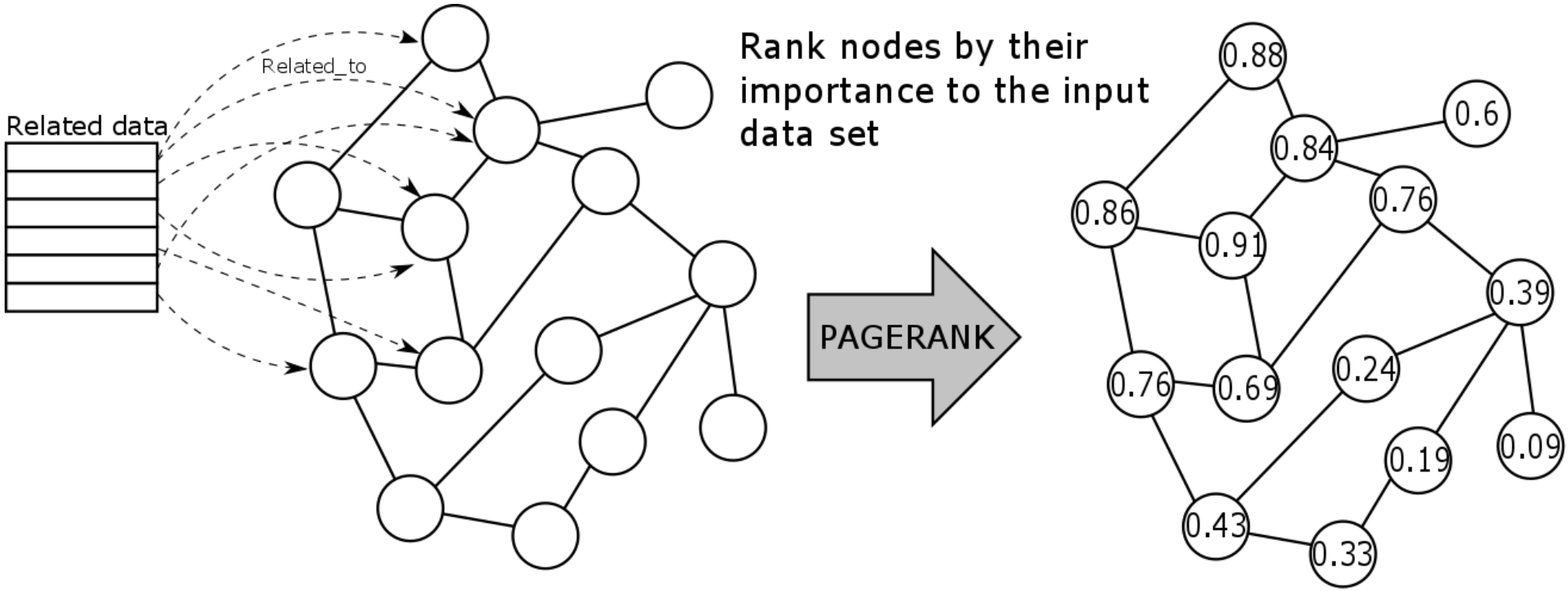
- Take a large knowledge graph such as BioMine, or a Linked Open Data resource, such as Bio2RDF
- Use Semantic data mining (SDM) to mine experimental data with ontologies as background knowledge to get explanations for groups of TargetClass objects, e.g.

  BreastCancer ← chromosome AND cell cycle

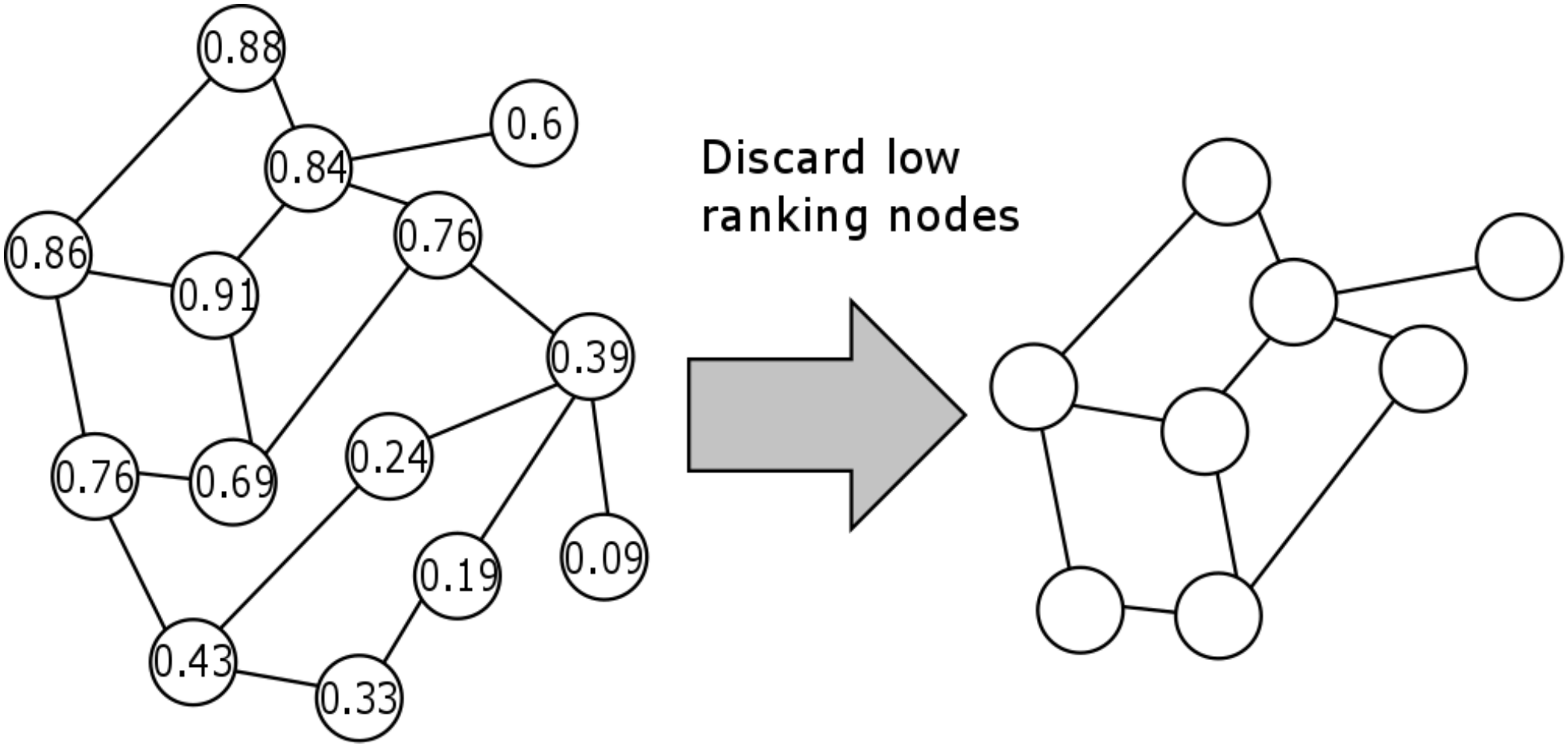
- Reduce the complexity of the huge search space of ontology terms by network analysis based node filtering

(Kralj et al., MLJ 2019)

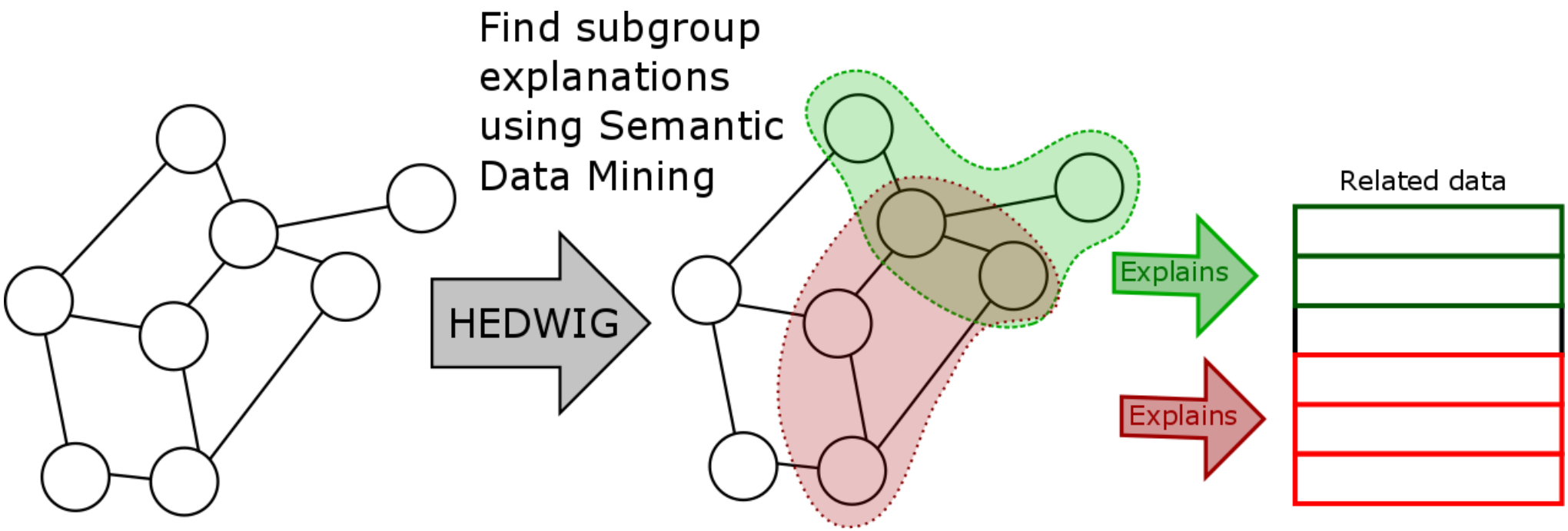
NetSDM Methodology: Step 1



NetSDM Methodology: Step 2

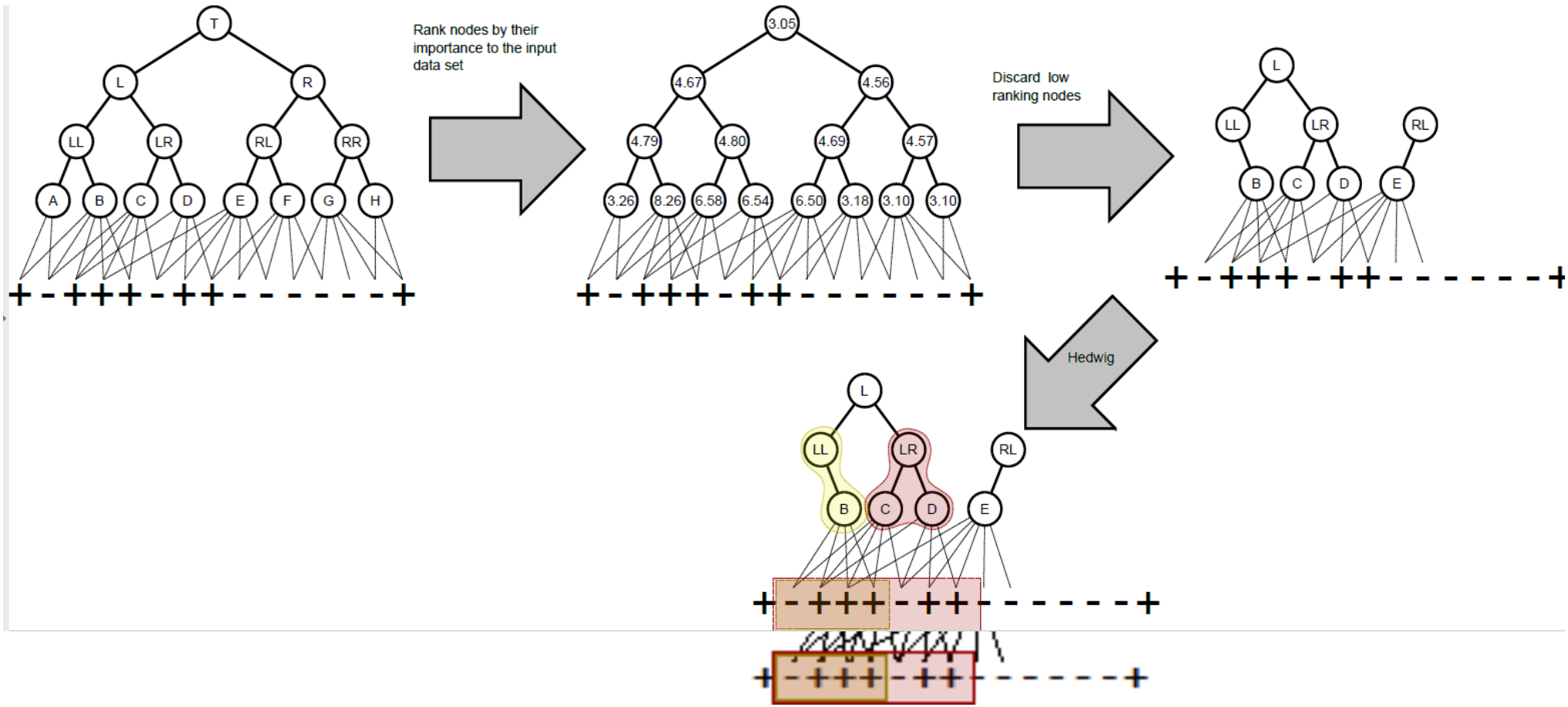


NetSDM Methodology: Step 3



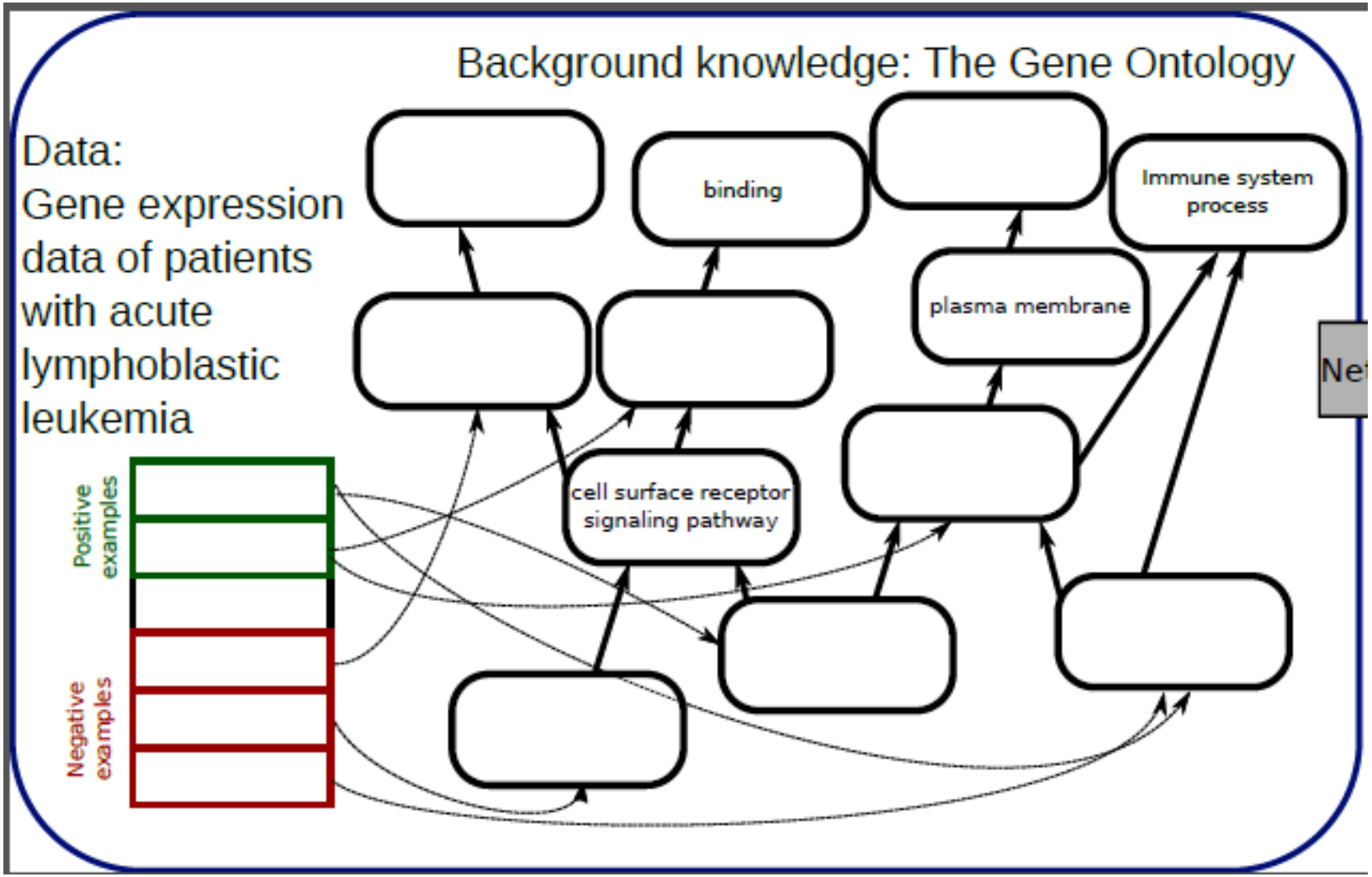
NetSDM algorithm outline

1. Estimate ontology term relevance
2. Delete terms with low relevance
3. Run Hedwig on pruned ontolgy



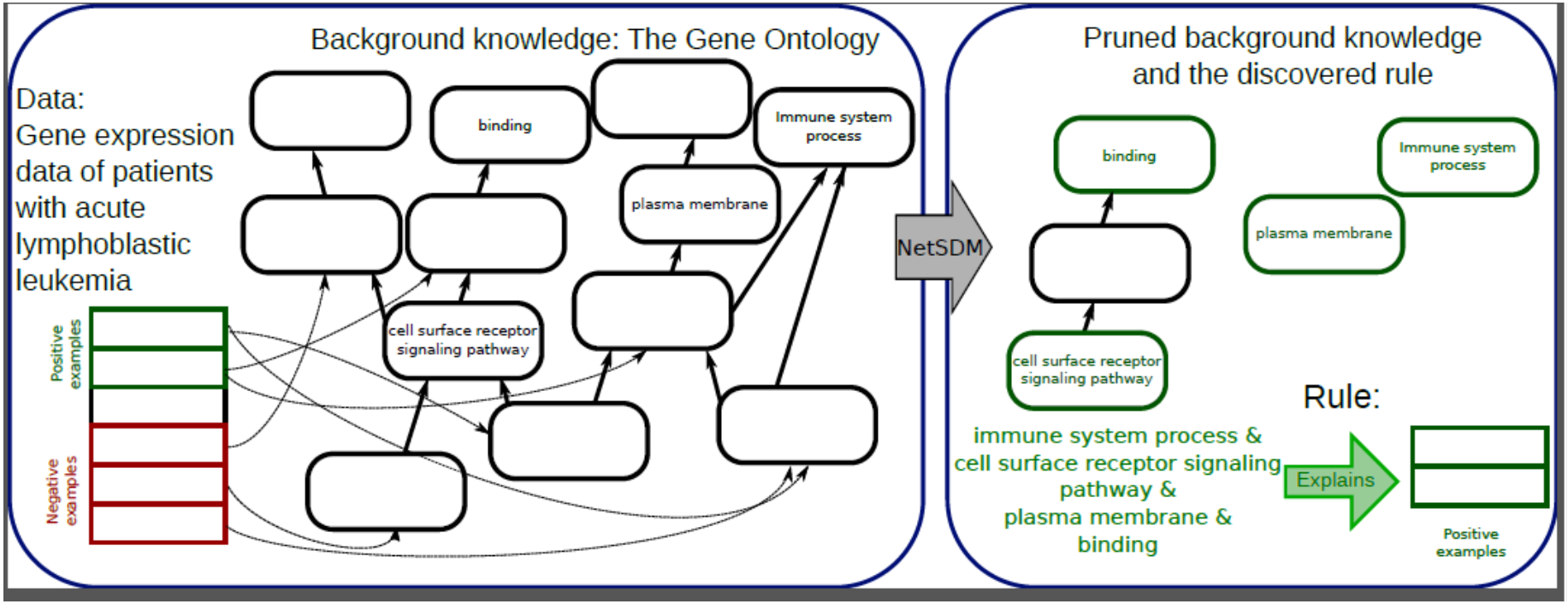
Example: Analysis of ALL data using Gene Ontology

Input to NetSDM:



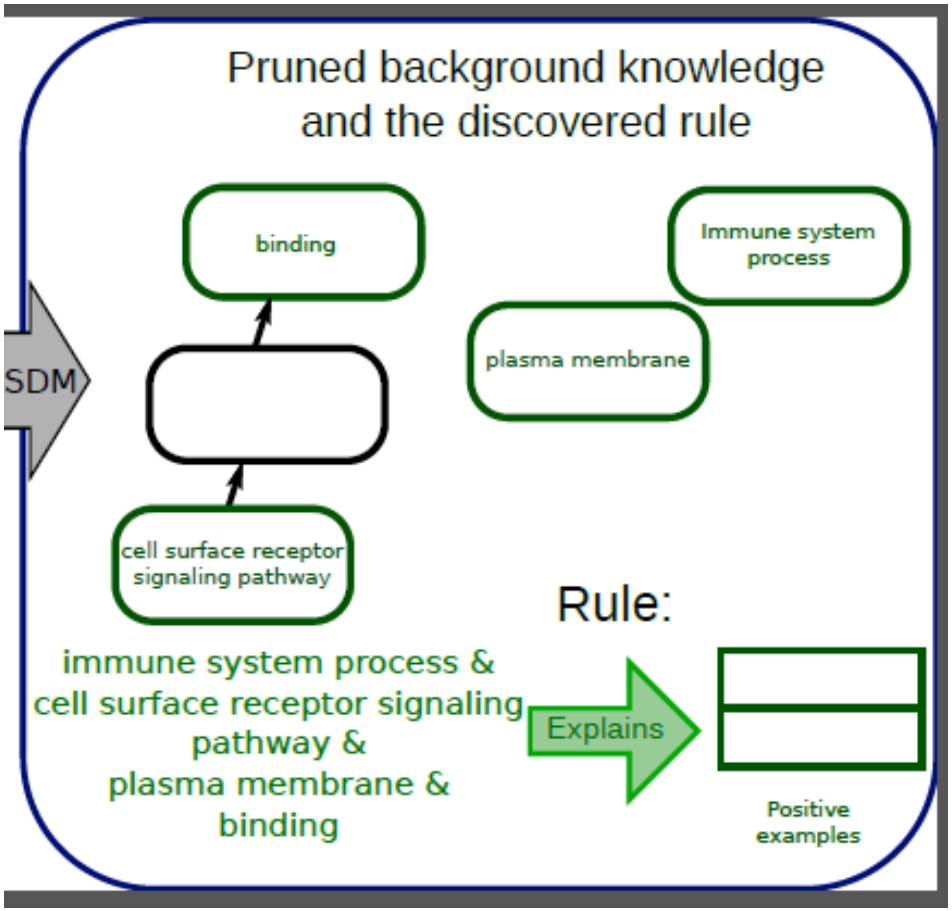
Example: Analysis of ALL data using Gene Ontology

NetSDM:



Example: Analysis of ALL data using Gene Ontology

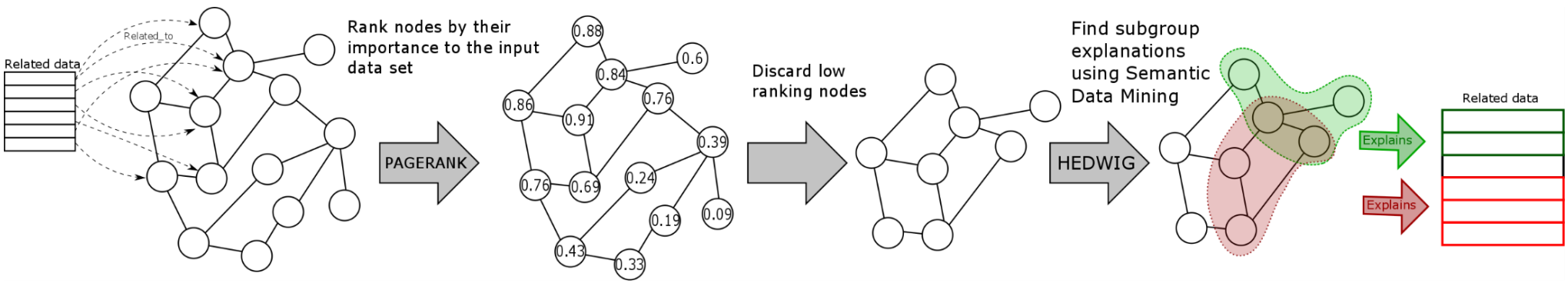
Output of NetSDM:



NetSDM Results

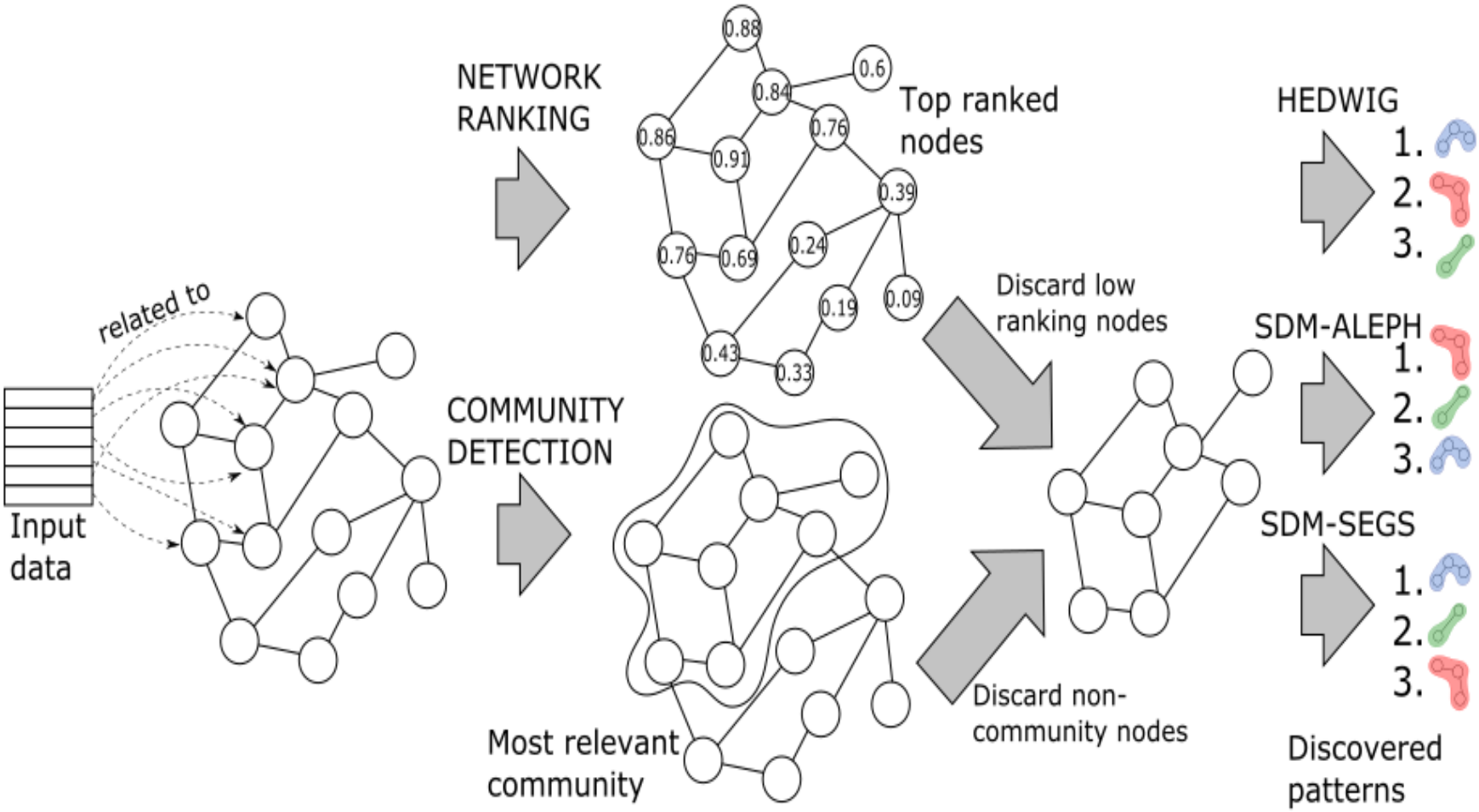
- PageRank can be effectively used to decrease the size of the search space of Semantic Data Mining algorithms
- Accuracy did not decrease even when significantly decreasing the size of the background knowledge to less than 5%.
- Time, taken to discover rules on pruned background knowledge, is shorted by a factor of 100

(Kralj et al. 2017)



NetSDM methodological framework

Network based approaches such as ranking and community detection are first used to extract relevant networks, and SDM algorithms (such as Hedwig or SDM-Aleph) may then discover patterns in the input data



SDM in context