Einblicke ins Dickicht der Parteiprogramme

Michael Hunger
Director Developer Relations Neo4j

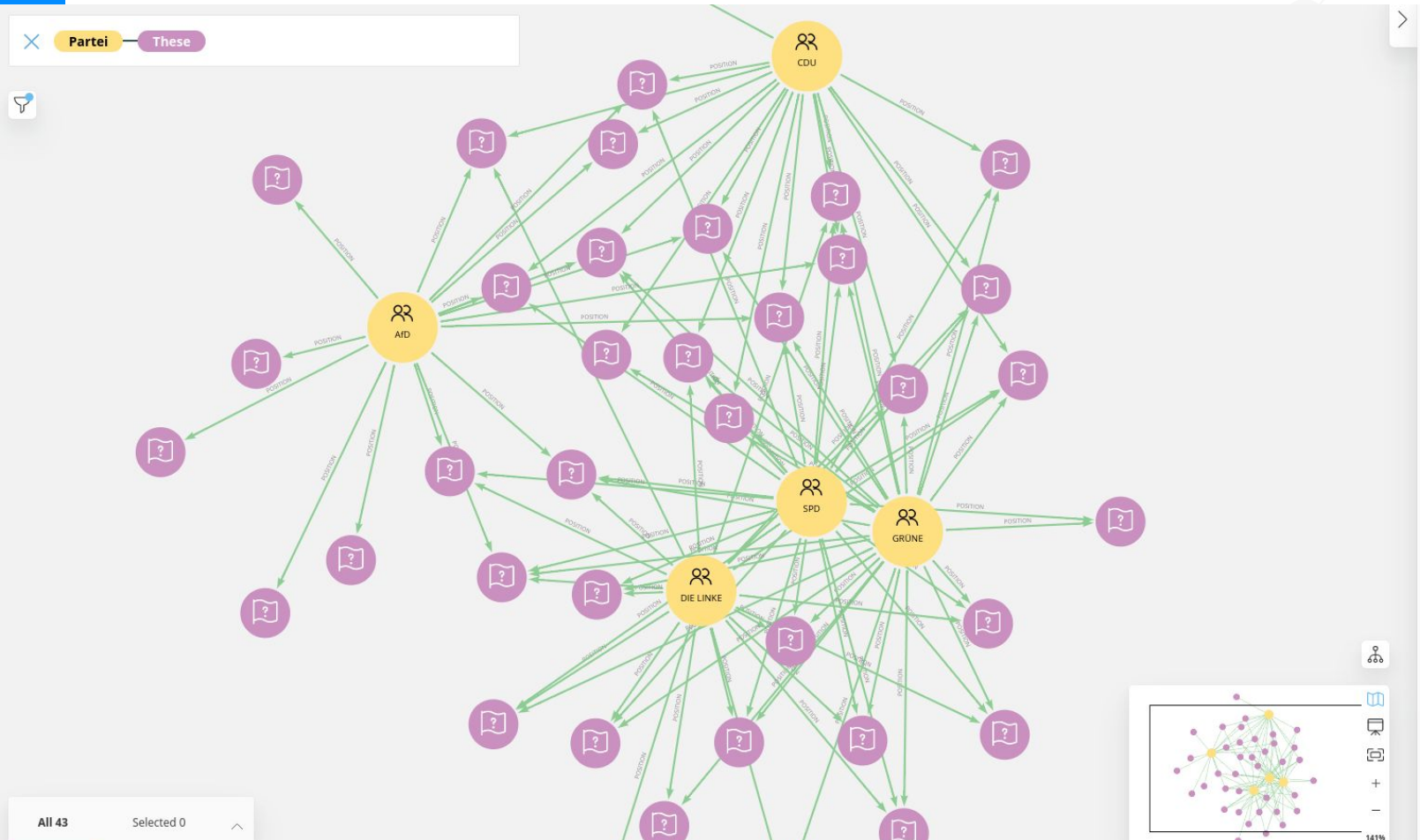
dev.neo4j.com/ltw-sa21

neo4j

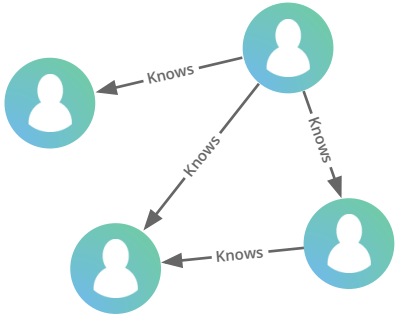
Es ist Wahljahr #btw2021 - Und wir alle gehen hin!

Wahl-O-Mat® - Parteiprogramme im Vergleich

Visuelle und Algorithmische Analyse



Connections in Data are as Valuable as the Data Itself

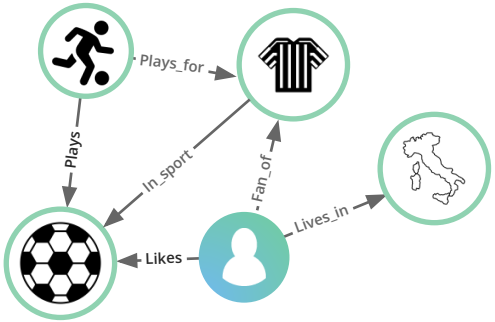


Networks of People

E.g., Employees, Customers, Suppliers, Partners, Influencers

Transaction Networks

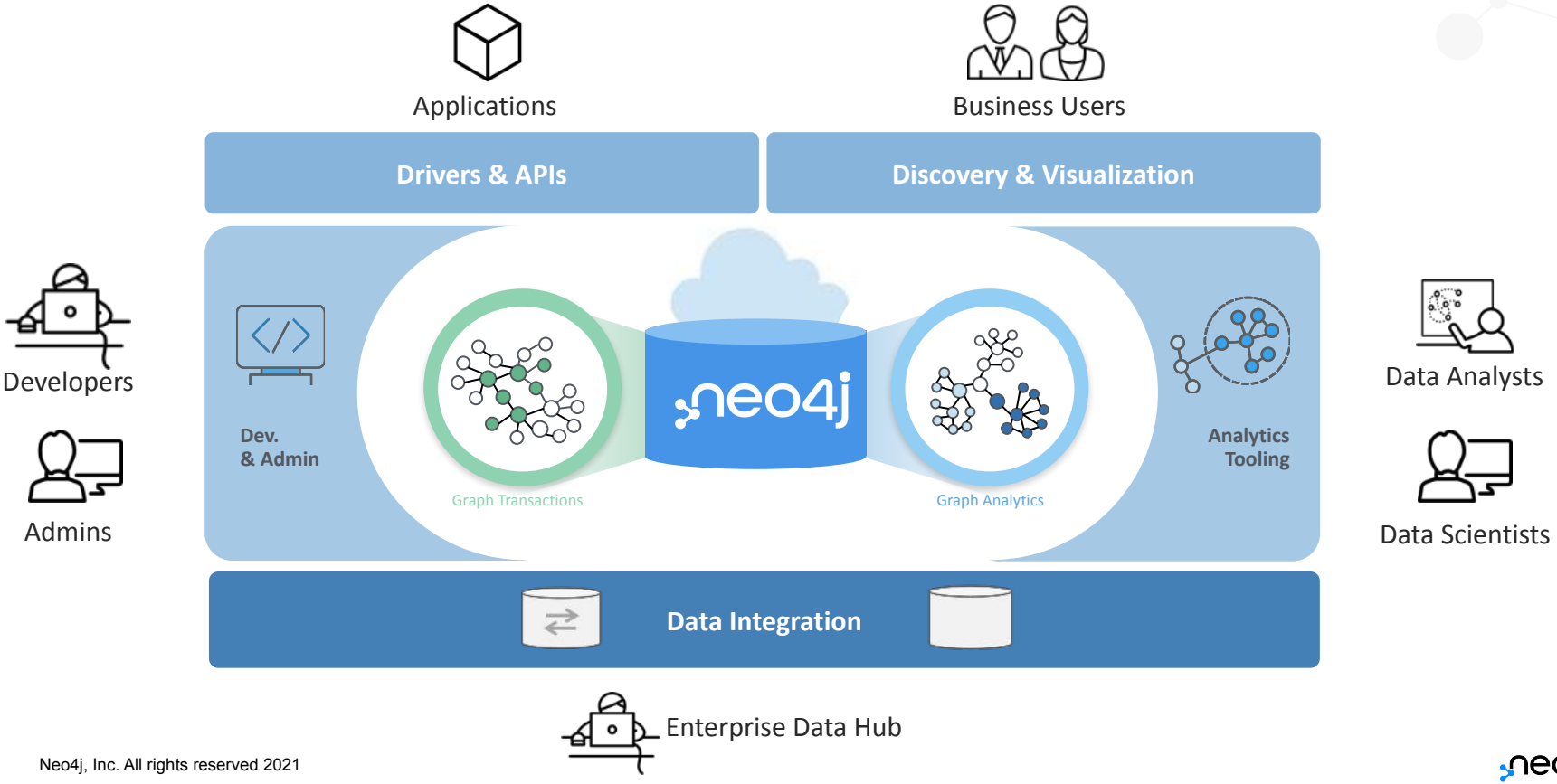
E.g., Risk management, Supply chain, Payments



Knowledge Networks

E.g., Enterprise content, Domain specific content, eCommerce content

Native Graph Technology for Applications & Analytics



Daten verfügbar!

Noch nicht für #btw21 aber für LTW Sachsen Anhalt

Download im CSV Format

bpb.de/politik/wahlen/wahl-o-mat/332469/download

ZIP mit CSV Dateien

Für einfachen Import auf GitHub

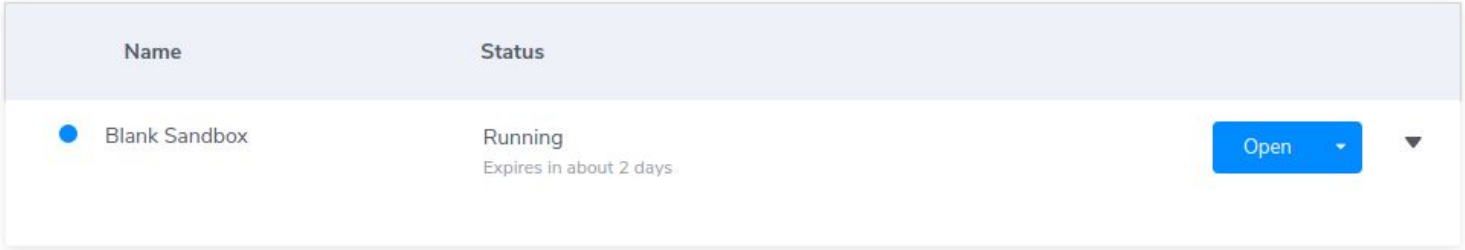
dev.neo4j.com/ltw-sa21

Import in sandbox.neo4j.com

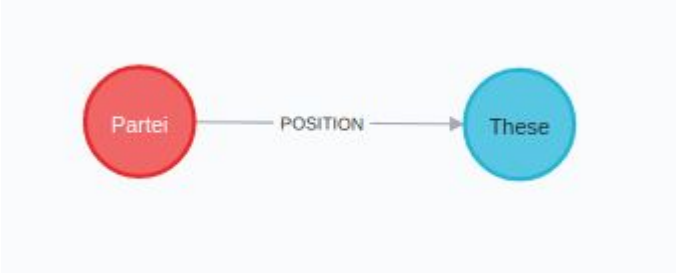
CSV Format

- Partei
  - Nr
  - **Name**
  - Kurzbezeichnung
- These
  - Nr
  - **Titel**
  - Text
- Position
  - **Position - stimme zu, neutral, stimme nicht zu**
  - Begründung

Import



- Knoten
  - Partei (21) (id, name, text)
  - These (38) (id, name, text)
- Beziehung
  - POSITION (798) (text, weight: 0.0-Ablehnung, 0.5-Neutral, 1.0-Zustimmung)
- via LOAD CSV + MERGE + CASE (siehe Script)
- Index on Parteiname/Thesenname



Exploration mit Neo4j Browser + Bloom

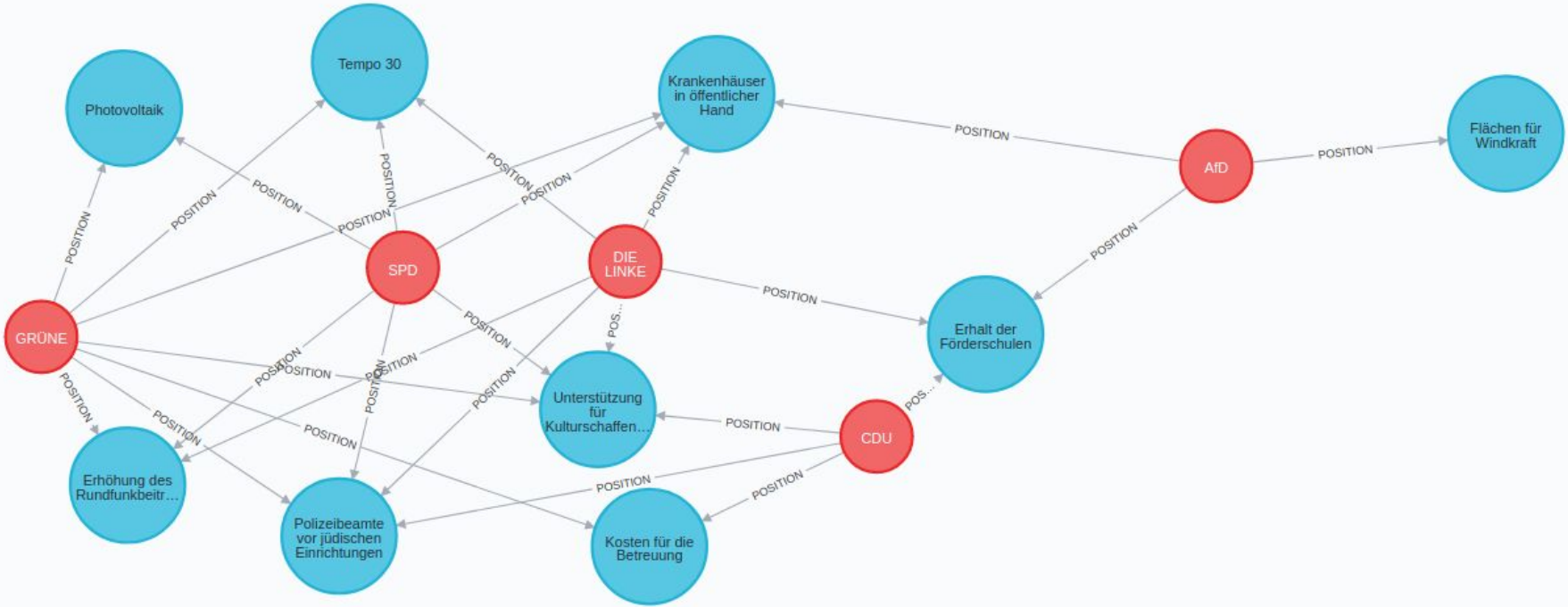
Abfragen + Visuelles Clustering

Exploration in Neo4j Browser + Bloom

```
$ MATCH (p:Partei)-[r:POSITION]→(t) where p.id ⩽ 5 and t.id <10 and r.weight = 1 RETURN *;
```

*(14) Partei(5) These(9)

*(26) POSITION(26)

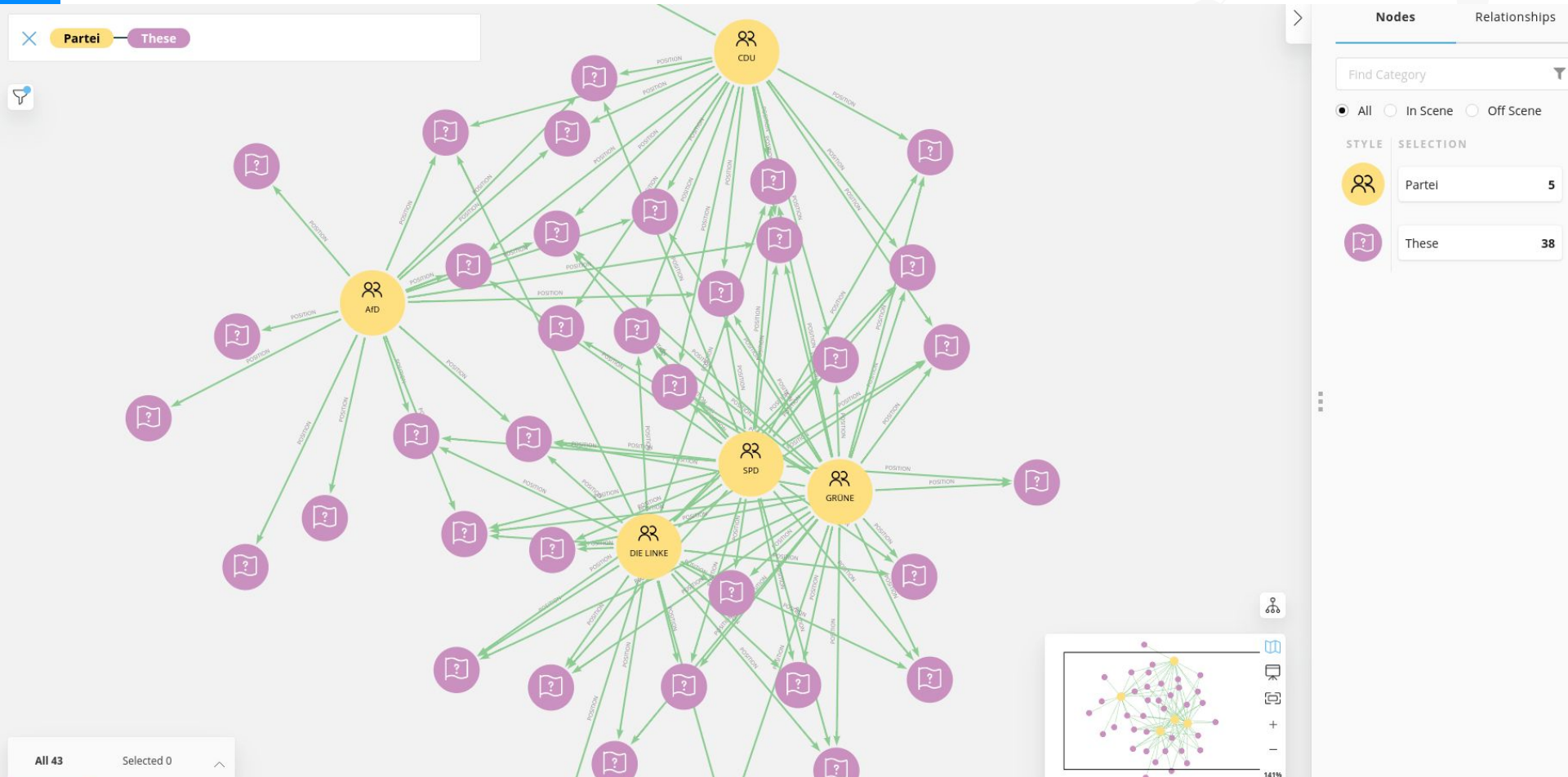


Ähnlichkeit von Parteien - "Abstand"

```
match (p1:Partei)-[r1:POSITION]->(t:These)<-[r2:POSITION]-(p2:Partei)
where id(p1)>id(p2)
return p1.name,p2.name, sum(abs(r1.weight-r2.weight)) as sim
order by sim asc;
```

"p1.name"	"p2.name"	"sim"
"Tierschutzallianz"	"Tierschutzpartei"	6.0
"PIRATEN"	"Tierschutzallianz"	6.0
"WiR2020"	"dieBasis"	6.0
"Tierschutzpartei"	"GRÜNE"	6.5
"Die PARTEI"	"GRÜNE"	6.5
"Die PARTEI"	"NPD"	26.0
"Die PARTEI"	"AfD"	26.5
"Klimaliste ST"	"AfD"	26.5
"LKR"	"DIE LINKE"	26.5
"Klimaliste ST"	"LKR"	27.0
"DIE LINKE"	"AfD"	28.0
"GRÜNE"	"AfD"	29.0

Exploration in Neo4j Browser + Bloom



Graph Data Science

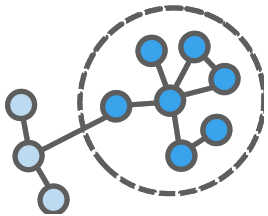
Abfragen + Visuelles Clustering

Neo4j for Graph Data Science™

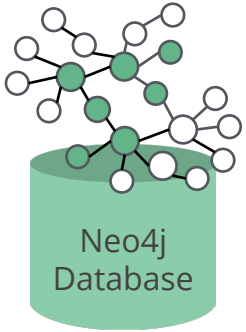
Scalable Graph Algorithms & Analytics Workspace

Native Graph Creation & Persistence

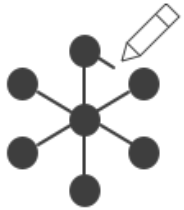
Visual Graph Exploration & Prototyping



Neo4j Graph Data Science Library



Neo4j Database



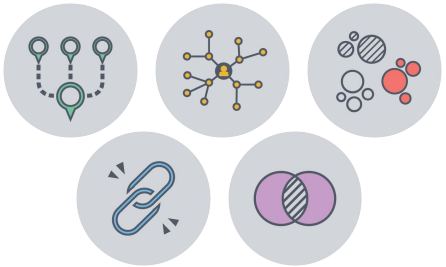
Neo4j Bloom

Practical

Integrated

Intuitive

The Neo4j GDS Library

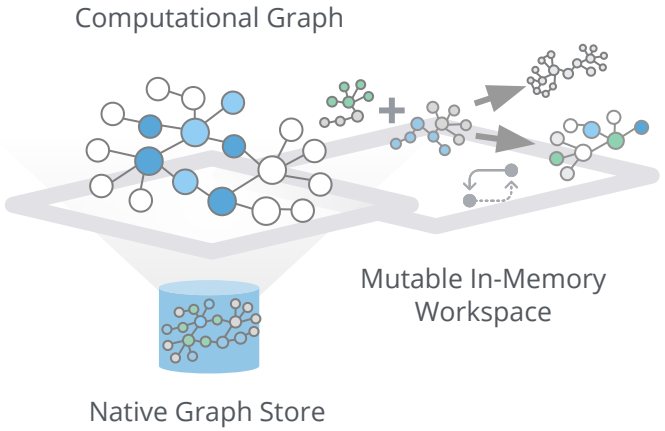


Robust Graph Algorithms (50+)

- Run on the loaded graph to compute metrics about the topology and connectivity
- Highly parallelized and scale to 10's of billions of nodes

Efficient & Flexible Analytics Workspace

- Automatically reshapes transactional graphs into an in-memory analytics graph
- Optimized for analytics with global traversals and aggregation
- Create workflows and layer algorithms



Projected Graphs

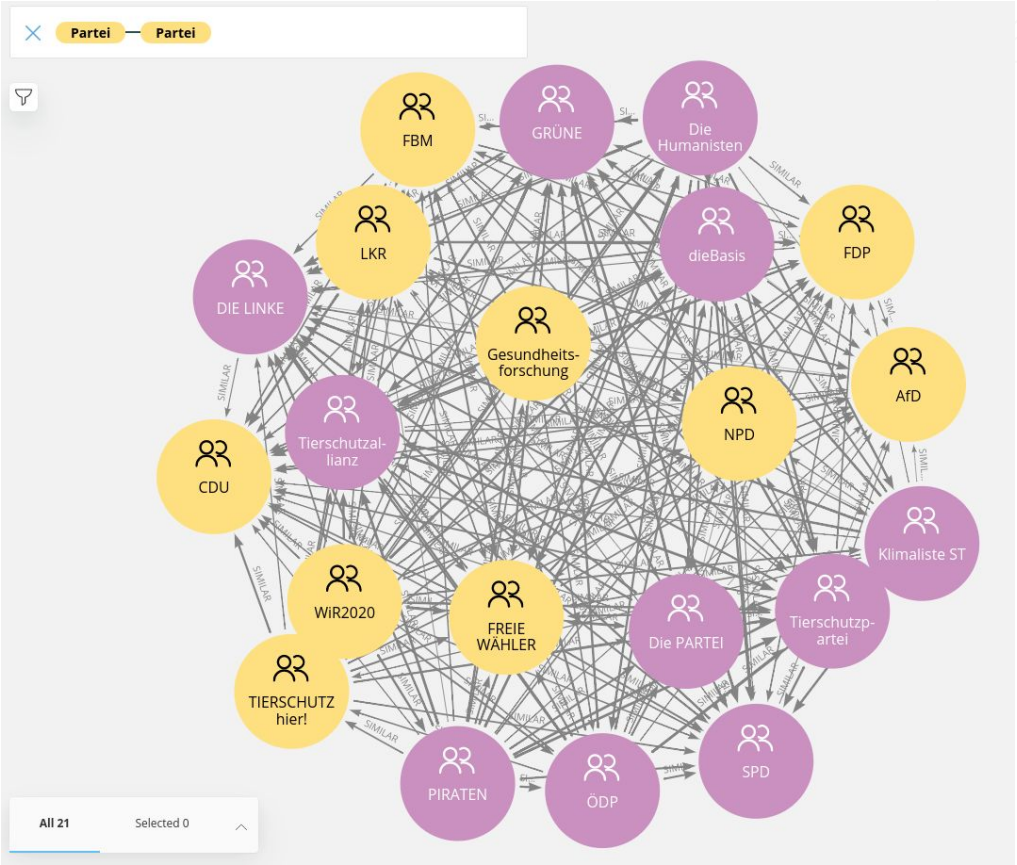
- (Partei) - [:SIMILAR {weight}] - (Partei)
  - virtuell in in-memory graph
  - physisch als relationship
  - "Distanz" als Gewicht normalisiert auf 0..1
- Clustering -> Louvain
- ML Model -> Link Prediction zur Thesis

Projection - virtuell

```
call gds.graph.create.cypher("parteien",
"MATCH (p:Partei) RETURN id(p) as id",
"MATCH
(p1:Partei)-[r1:POSITION]->(t:These)<-[r2:POSITION]-(p2:Partei)
 RETURN id(p1) as source,id(p2) as target,
(29.0-sum(abs(r1.weight-r2.weight)))/29.0 as weight");
```

Louvain

- Louvain Algorithmus
- Clustering mit Gewicht
- 2 cluster
- Visualisierung mit Bloom

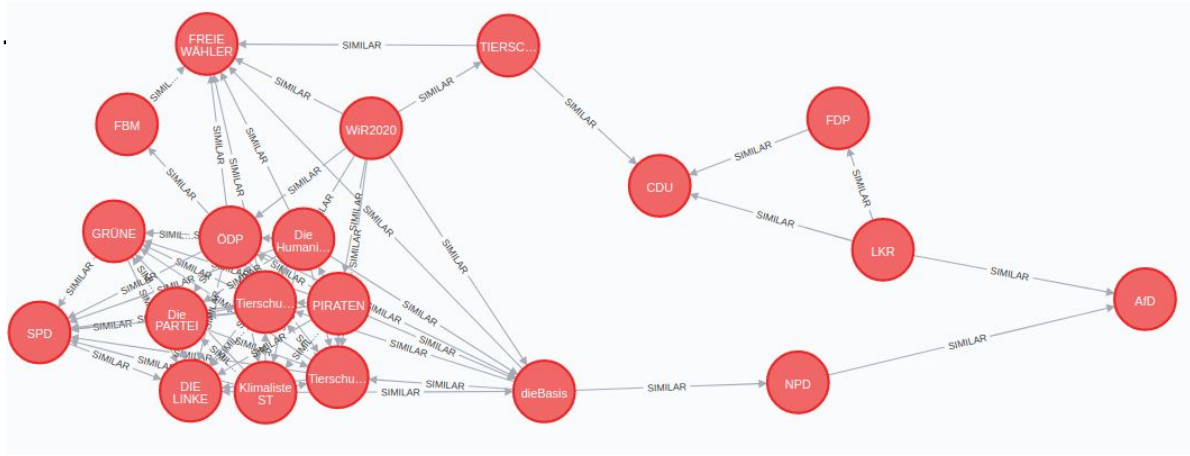



```
call gds.louvain.write("parteien",
                {relationshipWeightProperty:'weight', writeProperty:'cluster'})
```

Projection - phyische Beziehung

```
MATCH
(p1:Partei)-[r1:POSITION]->(t:These)<-[r2:POSITION]-(p2:Partei)
WHERE id(p1)>id(p2)
WITH p1,p2, (29.0-sum(abs(r1.weight-r2.weight)))/29.0 AS weight

MERGE (p1)-[s:SIMILAR]-(p2)
SET s.weight = weight
```



Node Embeddings & k-NN Similarity

- Node **Embeddings** -> Vektor repräsentiert "topologische Eigenschaften"
- kNN -> Ähnlichkeitsberechnung

1. load graph (undirected)
2. compute embedding (fastRP) + mutate in-memory graph
3. use embedding to compute k-NN topK (3) similarity

```
call gds.graph.create("p4",{Partei:{properties:"cluster"},These:{}},
                      {POSITION:{orientation:"UNDIRECTED",properties:"weight"}});

call gds.fastRP.mutate("p4",{relationshipWeightProperty:"weight"
                             embeddingDimension:128,
                             mutateProperty:"embedding"});

call gds.beta.knn.stream("p4",{nodeLabels:['Partei'],topK:3,
nodeWeightProperty:'embedding'})
yield node1, node2, similarity where node1 > node2
return gds.util.asNode(node1).name as n1,
          gds.util.asNode(node2).name as n2, similarity
order by similarity desc LIMIT 10;
```

n1.name	n2.name	similarity
"Die PARTEI"	"DIE LINKE"	0.9972781538963318
"Die Humanisten"	"SPD"	0.9969035387039185
"TIERSCHUTZ hier!"	"CDU"	0.9929470419883728
"FDP"	"CDU"	0.9924740791320801
"LKR"	"CDU"	0.9905626177787781
"AfD"	"CDU"	0.9597322344779968

Train ML Model & Predict Class (Cluster)

1. lade graph (undirected)
2. compute **embedding mit fastRP**
3. train model with embedding + class (cluster)
4. predict cluster

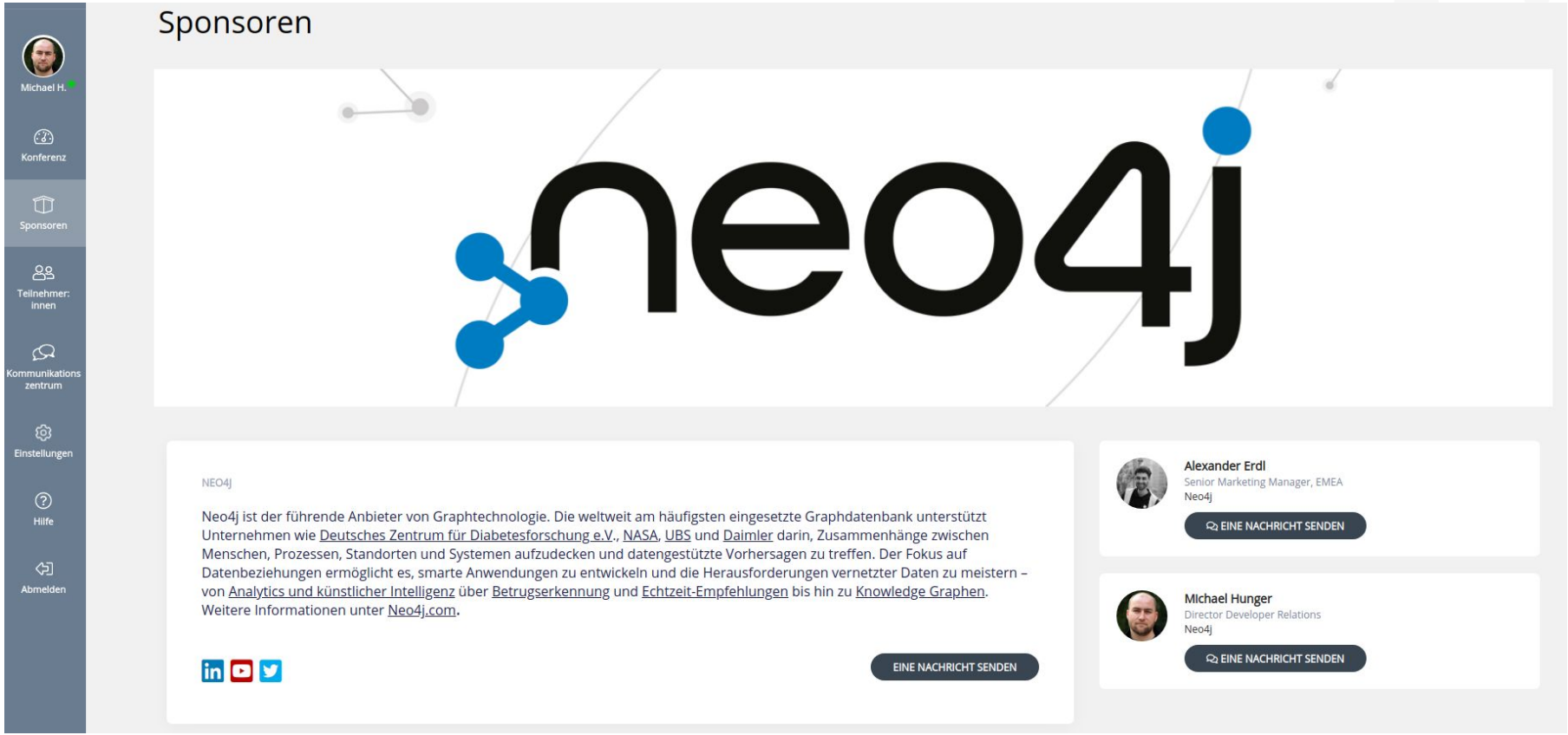
Keine tollen Ergebnisse,
Embedding muss verbessert
werden.

party	cluster	predictedClass	predictedProbabilities
"AfD"	10	15	[0.45584874719173485, 0.5441512528082647]
"DIE LINKE"	15	15	[0.4657041147182362, 0.5342958852817632]
"SPD"	15	15	[0.46059514148778885, 0.5394048585122106]
"GRÜNE"	15	15	[0.4896383093093259, 0.5103616906906735]
"FDP"	10	15	[0.48062074081041567, 0.5193792591895839]
"FREIE WÄHLER"	10	15	[0.4996539695593245, 0.500346030440675]

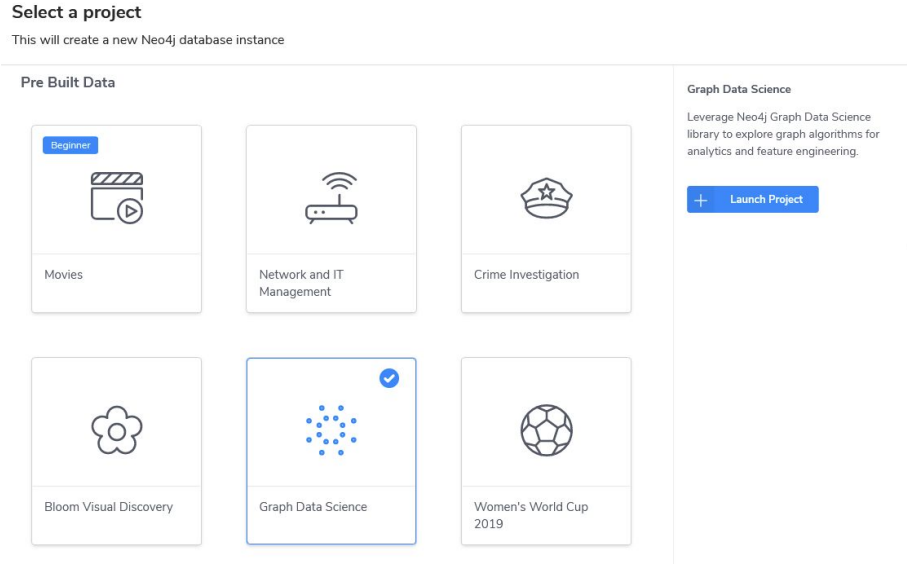
Weitere Ideen

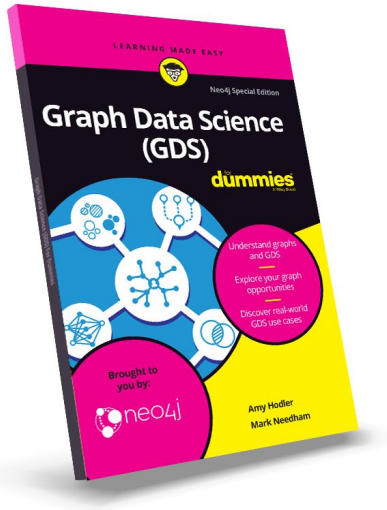
- eigene Position(en) hinzufügen und mit Parteien (und einander) vergleichen
- Veränderung von Positionen über Wahlperioden -> Bewegung der Schwerpunkte
- Korrelation mit Wahlergebnissen
- Positionen pro Demographie / Bevölkerungssegment
- Link Prediction / Node Classification

Fragen & Diskussionen ? Gern am Stand nachfragen!



Selbst ausprobieren?





sandbox.neo4j.com

dev.neo4j.com/gdsbk2

Danke für Ihre Aufmerksamkeit!

Michael Hunger
Director Developer Relations Neo4j

dev.neo4j.com/gdsl
sandbox.neo4j.com
dev.neo4j.com/ltw-sa21

Train ML Model & Predict Link

1. load graph (with PRO/CON links?)
2. split relationships into test / train & positive (exists) & negative (doesn't exist)
3. train model with existing links (works only on undirected graphs)
4. predict links

see GIST