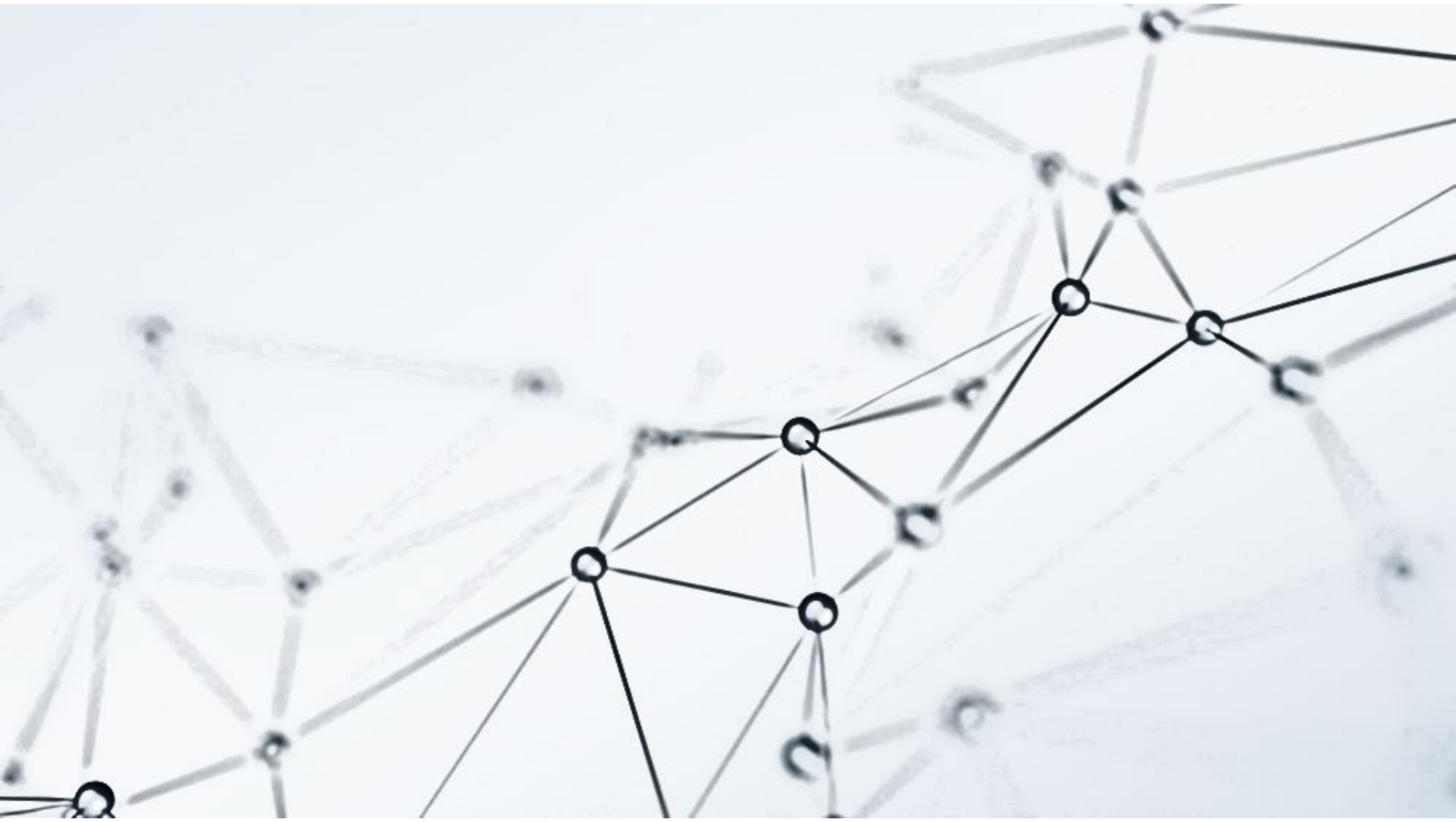
ZENTRUM FÜR KÜNSTLICHE INTELLIGENZ IN MV

KI MV



KNOWLEDGE GRAPHS

Achim Reiz
Zentrum für Künstliche Intelligenz in MV

Whitepaper-Serie des Zentrums für Künstliche Intelligenz in MV
Ausgabe 5

Titelbild Microsoft Stock Photos

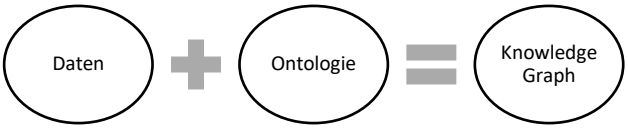
1 Einleitung

Daten sind das Gold des 21. Jahrhunderts. Mit diesem Bild im Hinterkopf wurden in den vergangenen Jahren enorme Datensammlungen angelegt. Die Speicherung alleine generiert jedoch noch keinen Mehrwert. Die Daten müssen auch verknüpft und integriert werden - das ist jedoch meist leichter gesagt als getan. Sie sind oft in heterogen, siloartigen Strukturen abgelegt. Oftmals gibt es eine Vielzahl an (Alt-)Systemen, die sich nur schwer integrieren lassen[1].

Knowledge Graphs (KGs, deutsch: Wissensgraphen, oft auch semantische Netze oder regelbasierte KI) bieten eine Lösungsmöglichkeit um dieser Komplexität Herr zu werden. Sie vernetzen die Daten in einer Graphstruktur, bestehend aus mit einander verbundenen Datenpunkten. Auf diese Weise lassen sich auch komplexe Sachverhalte modellieren. Auch bietet die graphbasierte Datenhaltung erhöhte Flexibilität – die Integration neuer Inhalte benötigt, im Gegensatz zu klassischen Datenbanksystemen, keine aufwendige Anpassung der Datenbankstruktur.

Zwar benötigen Knowledge Graphs kein starres Schema für die Datenspeicherung – gleichwohl bieten sie Möglichkeiten der Datenstrukturierung. Über formale Regeln können wir dem Computer die Zusammenhänge in den Daten erklären. Auf diese Weise lassen sich Integrationsregeln und Querverbindungen zwischen den verschiedenen Datenstrukturen hinterlegen, auch Geschäftsregeln können modelliert werden. Die Gesamtheit dieser Regeln nennt sich *Ontologie*.

Die Kombination aus Ontologie und Daten schafft nun neue Möglichkeiten: Je nach definierten Regeln lassen sich die Daten auf Korrektheit und Vollständigkeit überprüfen (z.B.: Jedes Kind hat zwei biologische Eltern) oder unausgesprochenes, implizites Wissen automatisch erkennen (z.B. Der Vater eines Vaters ist ein Großvater). Daher werden Knowledge Graphs auch als regelbasierte oder deklarative künstliche Intelligenz bezeichnet.



2 Die Grundbausteine

Die Kernbausteine von KGs sind durch weitestgehend durch das World Wide Web Consortium (W3C) standardisiert. Die Standardisierung ermöglicht ein hohes Maß an Interoperabilität zwischen verschiedenen Anwendungen und Services: Daten lassen sich unkompliziert zwischen verschiedenen Anbietern und Diensten transportieren, das Risiko durch einen Anbieterausfall ist minimiert.

2.1 Die Daten

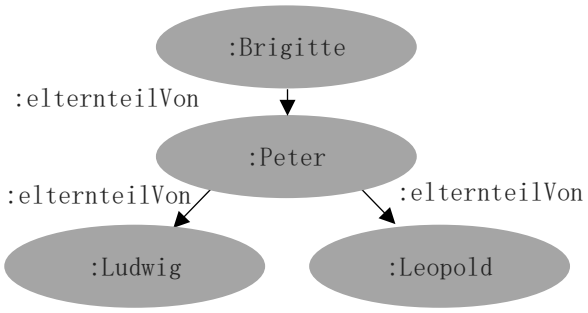
Der erste Grundbaustein der Knowledge Graphs definiert, wie der Graph gebildet wird. Das *Resource Description Framework (RDF)* beschreibt das Ablegen von Informationen in einer *Subjekt Prädikat Objekt* Struktur, zusammen auch *Triple* oder *Axiom* genannt. Die einzelnen Ressourcen sind hierbei stets *Unique Resource Identifiers* (URI). Eine URI sieht oft aus wie eine URL, bekannt aus dem World Wide Web. Im Gegensatz zu einer URL (*Unique Resource Locator*) steht hinter einer URI jedoch nicht zwangsweise eine lesbare Online-Ressource. Ihre Kernaufgabe ist die eindeutige Identifikation. So ist jede URL auch eine URI. Auch andere eindeutige Identifizierungen sind gültige URIs, z.B. ISBN-Nummern. RDF modelliert nun Zusammenhänge zwischen diesen eindeutig identifizierten Ressourcen.

Stellen wir vor, wir wollen eine Familienstruktur modellieren. *Brigitte* ist ein Elternteil von *Peter* und *Peter* ist das Elternteil von *Ludwig* und *Leopold*. In RDF modelliert sehen diese Statements folgendermaßen aus:

```
@prefix kimv: <http://www.ki-mv.de/family#>.
kimv:Brigitte kimv:elternteilVon  kimv:Peter .
kimv:Peter  kimv:elternteilVon  kimv:Ludwig .
kimv:Peter  kimv:elternteilVon  kimv:Leopold .
```

Die erste Zeile definiert einen Namensraum (englisch: Namespace) und dient der besseren Lesbarkeit. Das `kimv:` wird durch den Computer ersetzt durch die angegebene URI (z.B. `kimv:Brigitte` → `http://www.ki-mv.de/family#Brigitte`). Der Punkt markiert das Ende eines Triples.

Diese Triple bilden zusammengenommen eine Graphstruktur:



2.2 Die Ontologie

RDF erlaubt auf einfache Art und Weise, Elemente miteinander zu verknüpfen. Alle drei Statements, das *Subjekt* und *Objekt* als auch das verbindende *Prädikat* sind URIs, die eine Ressource eindeutig identifizieren.

Für den Computer haben diese selbsterstellten URIs jedoch keinerlei Bedeutung. In RDF abgelegtes Wissen lässt sich zwar strukturiert abfragen (siehe Abschnitt 2.3). Die Funktionalität geht jedoch nicht über das extrahieren der vorhandenen Daten hinaus und unterscheidet sich, abgesehen von der Struktur, kaum von etablierten Datenbanksystemen.

Der Schritt von Graphdaten hin zu symbolischer künstlicher Intelligenz erfolgt durch den Einsatz computerverständlicher Regeln über die Zusammenhänge der Daten, abgelegt in der *Ontologie*. Die Ontologie ist eine formale Beschreibung der für den Anwendungszweck relevanten Welt. Auf ihrer Basis kann der Computer z.B. prüfen, ob die Daten in ihrer Struktur oder Inhalt korrekt sind. Auch lassen sich weitere Informationen aus den Daten schließen.

Die Regeln nutzen ebenfalls die RDF-Struktur, verwenden jedoch besondere Ressourcen (also URIs). Ausgelegt für das Schließen von Informationen sind z.B. die *Web Ontology Language (OWL)* und *RDF Schema (RDFS).* Diese finden sich unter den immer gleichen URIs in dem *RDFS*- oder *OWL*-Namensraum. Die KG-Applikationen erkennen diese festgelegten Begrifflichkeiten und wissen diese zu interpretieren.

Nun aber genug der Theorie: Für die oben genannten Familiendaten können wir den Computer anhand von Regeln erklären, wie Familienbeziehungen zusammenhängen. Im Folgenden definieren wir erst die Verbindung für `kimv:elternteilVon`. Das erste Statement, `kimv:elternteilVon a owl:ObjectProperty`, definiert `kimv:elternteilVon` als eine Beziehung zwischen zwei Instanzen. Als Inverses wird eine Beziehung `kimv:istNachkommeVon` beschrieben. Das `a` ist die Kurzform für rdf:type, und beschreibt eine *„ist ein“*-Beziehung. `kimv:Peter`, `kimv:Brigitte`, `kimv:Ludwig` und `kimv:Leopold` definieren wir als Instanzen:

```
kimv:elternteilVon a owl:ObjectProperty .
kimv:elternteilVon owl:inverseOf kimv:istNach-
  kommeVon .
kimv:Peter a owl:NamedIndividual .
kimv:Ludwig a owl:NamedIndividual .
kimv:Leopold a owl:NamedIndividual .
kimv:Brigitte a owl:NamedIndividual .
```

Auf Basis der `kimv:elternteilVon` Beschreibung lassen sich weitere Beziehungen zwischen Elementen definieren. Die nächsten Statements definieren die Beziehung `kimv:großelternVon` als eine zweifache Verkettung der Beziehung `kimv:elternteilVon`. (Die Eltern der Eltern sind die Großeltern.) Eine ähnliche Verkettung nutzt die `kimv:geschwister` Beziehung. Diese ist definiert als die Verkettung

der Beziehung `kimv:istNachkommeVon` und `kimv:elternteilVon` (Ein Geschwister ist der Nachkomme des selben Elternteils.).

```
kimv:großelternVon a owl:ObjectProperty .
kimv:großelternVon owl:propertyChainAxiom (
  kimv:elternteilVon kimv:elternteilVon ) .
kimv:geschwister  a owl:ObjectProperty .
kimv:geschwister owl:propertyChainAxiom  (
  kimv:istNachkommeVon kimv:elternteilVon ) .
```

Die formale Beschreibung der Familienverhältnisse erlaubt einer Inferenzmaschine nun das automatisierte Schließen von impliziten (unausgesprochenen) Wissen. Der Computer erkennt die `owl:`-Keywords und weiß sie entsprechend zu interpretieren. Für die drei Datenpunkte in Abschnitt 2.1 kann sie eine Reihe weiterer Datenpunkte hinzufügen. Für Ludwig wird automatisch hinzugefügt:

```
kimv:Ludwig kimv:geschwister kimv:Leopold .
kimv:Ludwig :hatGroßeltern kimv:Brigitte .
```

Obwohl wir in den Daten nur festgelegt haben, dass Peter ein Elternteil von Ludwig ist, haben wir nun auch das Geschwisterkind und die Großeltern in den Daten explizit hinterlegt.

Neben der für das automatisierte Schließen ausgelegten `owl:` Ressource gibt es eine Vielzahl weiterer standardisierter Ontologien mit festgelegter Bedeutung, u.a.: die *Shape Constraint Language (SHACL). Sie* erlaubt das Überprüfen von Daten auf Korrektheit anhand syntaktischer Regeln. Das *Simple Knowledge Organisation System (SKOS)* beschreibt Taxonomien (hierarchische Begriffsklassifizierungen). Die *Provenance* (*PROV*) Ontologie bietet Begriffe für die strukturierte Beschreibung von Herkunftsinformationen (wer hat wann was geschrieben). *Dublin Core (DC)* beschreibt Metadaten.

Ziel einer Ontologie ist stets das Erzeugen eines gemeinsamen Begriffsverständnisses zwischen Computern und Menschen. Verwendet man z.B. das Prädikat `dc:license` aus der DC-Ontologie, so kann man die Lizenz einer Ressource menschen- und computerverständlich beschreiben – unabhängig davon, welcher Softwarehersteller zum Einsatz kommt. Mit unserer Familienontologie verhält es sich ähnlich: Die formale Beschreibung der Familienverhältnisse macht unsere Welt dem Computer und anderen Menschen verständlich.

2.3 Das Abfragen von Wissen

Wir haben nun unser Wissen in eine formale, elektronische Form überführt und im besten Falle auf einem Server für Knowledge Graphs, einem Triplestore, gespeichert. Nun benötigen wir jedoch auch eine Möglichkeit, das kodierte Wissen zu nutzen. Hierfür kommt die Abfragesprache SPARQL zum Einsatz.

Hiermit können wir die Graphstruktur unseres strukturierten Wissens abfragen. Die Abfragen erfolgen ebenfalls in Tripleform:

```
PREFIX kimv: <http://www.ki-mv.de/family#>
SELECT DISTINCT * {
?großEltern kimv:großelternVon kimv:Ludwig .
?eltern kimv:elternteilVon kimv:Ludwig }
```

In der Abfrage lassen sich teile der Triplestruktur durch Variablen ersetzen. Diese werden dann durch die eventuell vorhandenen Daten befüllt. Die hier dargestellte Abfrage liefert folgendes Ergebnis:

großeltern	eltern
Brigitte	Peter

Im Hintergrund sucht der Triplestore die Triple aus, die zu den in der Abfrage angegebenen Rahmenbedingungen passen und gibt die befüllten Variablen aus.

3 Die Stärken von strukturiertem Wissen

Die Erstellung und Pflege von Knowledge Graphs ist aufwendig. Zum einen birgt die Technologie selbst ein hohes Maß an Komplexität, insbesondere wenn Ontologien für das

Aufbrechen siloartiger Datenstrukturen, die automatische Datenüberprüfungen oder das Schließen von Wissen genutzt werden sollen.

Das wirft die Frage auf: Ist dieser vergleichsweise große Aufwand gerechtfertigt, wenn doch große Sprachmodelle wie ChatGPT (OpenAI) oder Bard (Google) einen natürlichsprachigen Zugriff auf Wissen ermöglichen?

3.1 Die Schwächen der Sprachmodelle

Sprachmodelle (Auch: Large Language Models, LLM) basieren auf den Methoden des maschinellen Lernens. Vereinfacht ausgedrückt geben sie die statistisch am besten passenden Wörter aus, die in dem gegebenen Fragekontext passen. Der Nutzer hat jedoch keinen Einblick, warum das Modell genau diese Ausgabe errechnet hat. Oftmals sind diese Antworten jedoch korrekt: Die Frage nach der Einwohnerzahl Rostocks wird von ChatGPT (GPT-3.5) richtig beantwortet, auch die Antwort auf die Frage nach dem Alter der Petrikirche beantwortet das Sprachmodell korrekt. Wir können es jedoch schnell an seine Grenzen bringen:

Die Frage nach der aktuellen Bürgermeisterin scheitert aufgrund der älteren Datenlage. ChatGPT gibt den vorherigen Oberbürgermeister an. Die Frage nach den letzten vier Bürgermeistern und deren Alter bei Amtsantritt verweigert das Sprachmodell jedoch völlig.

Die beiden Beispiele zeigen (Stand 2023) die Grenzen der Sprachmodelle auf. Diese werden auf Basis der Datenlage eines bestimmten Stichtages trainiert und neue Daten kommen ausschließlich in einem erneuten, äußerst aufwendigen Training hinzu. Neuste Entwicklungen sind daher oftmals nicht Teil der Datenlage. Weiter kommen die Sprachmodelle an ihre Grenzen, wenn Faktenwissen miteinander kombiniert werden muss. Hier erfindet (halluziniert) das LLM teilweise sogar Fakten, die nicht korrekt sind.

3.2 Knowledge Graphs vs. LLMs

Knowledge Graphs sind gewissermaßen das Gegenteil der Sprachmodelle: Sie bieten kontrolliertes Wissen: Es lässt sich zu jeder Zeit nachvollziehen, warum eine Anfrage genau diese Ausgabe generiert. Die Datenbasis lässt sich schnell anpassen. Weiter erlauben Knowledge Graphs die Verknüpfung verschiedenster, komplexer Informationen.

Das zeigt der Vergleich zwischen ChatGPT und Wikidata. Wikidata ist die Wikipedia abgebildet als Knowledge Graph. Das Wissen der Welt ist hier auf maschinenlesbare Art und Weise miteinander verknüpft. Im Folgenden sehen wir das Beispiel von Abschnitt 3.1 als SPARQL-Abfrage an die strukturierte Enzyklopädie.

```
SELECT DISTINCT ?majorName ?age {
  wd:Q2861 p:P6 ?heads.
  ?heads ps:P6 ?major;
    pq:P580 ?startTime.
  ?major rdfs:label ?majorName ;
         wdt:P569 ?birthdate
  BIND((?startTime - ?birthdate)/365 AS ?age)
  FILTER((LANG(?majorName)) = "de")
}ORDER BY DESC (?startTime ) LIMIT 4
```

Zum einen fällt auf: Die Anfrage ist offensichtlich deutlich komplexer als eine natürlichsprachige Anfrage an ein LLM. WikiData nutzt zudem vergleichsweise kryptische Abkürzungen für ihre URIs: `wd:Q2861` steht für Rostock, `p:P6` identifiziert eine Liste der Amtszeiten, welche u.a. einen Bürgermeister (`ps:P6`) und einen Start (`pq:P580`) zugeordnet haben.

Das Ergebnis der strukturierten Anfrage an eine kontrollierte Wissensbasis kann sich jedoch sehen lassen: WikiData identifiziert korrekt die letzten vier Bürgermeister mit einer genauen Altersangabe bei Amtsantritt.

majorName	age
Eva-Maria Kröger	40.658
Claus Ruhe Madsen	46.378
Roland Methling	50.852
Ida Schillen	47.726

3.3 Knowledge Graphs im Unternehmen

Die verknüpften Daten lassen sich in vielfältigen Unternehmensanwendungen verwenden. Z.B. nutzt Bosch Knowledge Graphs, um große Daten aus automatischen Schweißsystemen zu verknüpfen und ermöglicht auf diese Weise eine Qualitätsüberwachung[2], der französische Mobilfunkanbieter Orange verwendet Knowledge Graphs um komplexe IoT (Internet of Things)-Netzwerke zu managen[3] und die deutsche Bahn verbindet automatisiert erstellte Videodokumentationen mit Daten über die Bahninfrastruktur[4].

3.4 Knowledge Graphs in Kombination mit maschinellen Lernen

Semantische KI und Verfahren des maschinellen Lernens (ML) sind keine Technologien, die sich gegenseitig ausschließen. Die beiden Ansätze lassen sich zu hybrider (auch: neurosymbolischer) KI verknüpfen, um die Vorteile aus beiden Welten zu nutzen und die jeweiligen Nachteile zu mitigieren.

Der Kreativität sind hier kaum Grenzen gesetzt: So können wir die strukturierten Daten mit SPARQL abfragen und hiermit ML-Modelle trainieren. Knowledge Graphs können strukturierte Metadaten für die Ablage von unstrukturierten Trainingsdaten enthalten. Die Ergebnisse aus einer ML-Detektion lassen sich durch einen Knowledge Graph absichern.

Zwei praktische Beispiele: Erkennt z.B. die ML-Einheit eines autonomen Fahrzeugs fälschlicherweise ein Stoppschild auf der Mitte der Autobahn, so kann der Knowledge Graph das Weltwissen bereitstellen, das Stoppschilder nicht auf Autobahnen stehen und eine Vollbremsung verhindern. Auch kann ein LLM bei der Erstellung der SPARQL-Abfragen unterstützen und die SPARQL-Ergebnisse in natürliche Sprache verwandeln.

4 Fazit

Knowledge Graphs verknüpfen Daten mit Wissen über diese Daten. Auf diese Weise sorgen sie für ein gemeinsames Begriffsverständnis zwischen Maschinen und Menschen. Dies macht Datenquellen interoperabel und erlaubt z.B. das automatisierte Schließen von Wissen oder das Überprüfen von Datenregeln. Sie bieten eine kontrollierte Wissensbasis. Wir können sicherstellen, dass die Faktenbasis aktuell und richtig ist.
Knowledge Graphs stehen nicht in Konkurrenz zu anderen KI-Verfahren, sondern lassen sich sowohl isoliert als auch in beliebigen Kombinationen mit anderen Verfahren nutzen. Auf diese Weise sind sie ein Baustein, um das volle Potential aus den eigenen Daten auszuschöpfen.

Das Zentrum für Künstliche Intelligenz wird vom Wirtschaftsministerium des Landes MV und dem Europäischen Fonds für regionale Entwicklung (EFRE) gefördert.

EUROPÄISCHE UNION
Europäischer Fonds für regionale Entwicklung

MV tut gut.

Kontakt

Emil Löffler
Zentrum für Künstliche Intelligenz in MV
Albert-Einstein-Straße 22, 18059 Rostock
emil.loeffler@uni-rostock.de

[1] A. Blumauer and H. Nagy, The knowledge graph cookbook: Recipes that work, 1st ed. Wien: edition mono/monochrom, 2020.

[2] B. Zhou *et al.*, "Scaling Data Science Solutions with Semantics and Machine Learning: Bosch Case"

[3] A. Guittoum, F. Aïssaoui, S. Bolle, F. Boyer, and N. de Palma, "Solving the IoT Cascading Failure Dilemma Using a Semantic Multi-agent System"

[4] K. Herbst, R. Krieg, and D. Friedenberger, "Railway track video Knowledge Base"