Integration von KI-Algorithmen in Umweltinformationssysteme mittels SensorThings API

Philipp Hertweck[1], Hylke van der Schaaf[1], Desiree Hilbring[1], Jonas Weis[2], Tanja Liesch[2], Matthias Budde[3]

Abstract: Künstliche Intelligenz oder Maschinelles Lernen wird zunehmend in der Umweltdomäne eingesetzt. Eine offene Frage dabei ist, wie entwickelte Algorithmen flexibel in Umweltinformationsysteme integriert werden können. Dieser Artikel befasst sich mit dieser Frage und untersucht den Einsatz des offenen Standards SensorThings API des Open Geospatial Consortiums (OGC) für die Integration von KI-Algorithmen in eine Geodaten-Infrastruktur. Die entwickelte Methode trägt mittels Container-Technologie dem Einsatz unterschiedlicher Technologien und unabhängigen Entwicklergruppen in verteilten Systemen Rechnung. Entwickelt und Erprobt wird diese Methode im Projekt NiMo 4.0. Hierfür dient als Beispiel ein Prognose-Algorithmus für die räumliche Vorhersage von Nitrat-Daten.

Keywords: Nitratmonitoring; Umweltinformatik; SensorThings API; Künstliche Intelligenz; Algorithmenintegration; Container; Docker

1 Einleitung

In den letzten Jahrzehnten sind in den Bereichen Verwaltung, Forschung, Wirtschaft und für die Öffentlichkeit Umweltinformationssysteme (UIS) mit vielfältigen Arten von Daten für unterschiedliche Zwecke entstanden. Weiterhin ergeben sich mit den Entwicklungen im Forschungsgebiet der Künstlichen Intelligenz (KI) neue Analysemöglichkeiten für die Daten dieser UIS. Eine Integration dieser KI-Algorithmen in etablierte und bestehende UIS bringen Herausforderungen mit sich. Bestehende Umweltinformationssysteme auf Basis von Umweltdatenbanken sind häufig mit proprietären Schnittstellen für spezifische Zwecke ausgerüstet. Eine region- und domänenübergreifende Analyse ist üblicherweise nur unter Beteiligung mehrerer Behörden oder Versorgungsbetriebe möglich. Zusätzliche erschwert wird dies durch die im föderalen System verteilten Zuständigkeiten. Im behördlichen Umfeld stammen teilweise bereits verfügbare offene Schnittstellen meist aus dem Umfeld der Geodateninfrastrukturen und tragen den INSPIRE Richtlinien mit der Umsetzung von Standards des Open Geospatial Consortiums Rechnung.

[1] Fraunhofer IOSB, Fraunhoferstr. 1, 76131 Karlsruhe, Deutschland philipp.hertweck@iosb.fraunhofer.de
[2] Karlsruher Institut für Technologie (KIT), Institut für Angewandte Geowissenschaften (AGW), Kaiserstr. 12, 76131 Karlsruhe, Deutschland
[3] Disy Informationssysteme GmbH, Ludwig-Erhard-Allee 6, 76131 Karlsruhe, Deutschland

Ziel ist daher eine leicht zu implementierende Schnittstelle für den Zugriff auf die notwendigen Quelldaten zu schaffen, um einerseits durch Wiederverwendbarkeit Synergien zu nutzen und andererseits um durch domänen- und zuständigkeitsübergreifende Analysen neue Erkenntnisse zu gewinnen. Darauf aufbauend können KI-Algorithmen in die vorhandenen UIS integriert werden. Sowohl bei der Datenbereitstellung als auch bei der Integration der Analysen soll auf offene Standards zurückgegriffen werden, um die Interoperabilität zu erhöhen.

2 NiMo 4.0

Im Rahmen des Forschungsprojektes Nitrat-Monitoring 4.0 – Intelligente Systeme zur nachhaltigen Reduzierung von Nitrat im Grundwasser (NiMo 4.0) [Li21] wird der Transfer von innovativen Lösungsansätzen von KI-Anwendungen im Grundwasser-Bereich von der universitären und institutionellen Forschung in die industrielle, anwendungsnahe Forschung und Praxis vorangetrieben. Übergeordnetes Ziel ist eine verbesserte Vorhersage von Nitrat im Grundwasser. Um die zu entwickelnden Vorhersagemechanismen und die dafür benötigten verschiedenen Datenquellen effektiv nutzen zu können, werden im Rahmen des Forschungsprojektes Fragestellungen aus IT-Sicht behandelt: z.B. die Bereitstellung und und Integration verschiedener Datenquellen. Hierzu zählen im Kontext des NiMo-Projektes seit Jahren aufgebaute Fachinformationssysteme und Umweltdatenbanken für die Gewässerüberwachung der Umweltverwaltungen und Wasserversorger, die aufgebaut wurden um die Wasserrahmenrichtlinie umzusetzen. Eine länderübergreifende Datenauswertung ist bisher nicht möglich. Ein Ziel in NiMo als Leuchtturmprojekt ist es, durch die Nutzung von offenen Standards, die Übertragbarkeit der Lösung von den Pilotregionen des Projektes auf weitere Regionen, sicherzustellen.

Anhand eines Anwendungsbeispiels aus dem NiMo 4.0-Projekt zeigt dieser Beitrag, wie mit Hilfe des leichtgewichtigen OGC Geodatenstandards SensorThings API, Daten aus einem bestehenden UIS bereitgestellt und durch einen KI-Algorithmus analysiert werden können. Dazu wird anhand einer KI-Nitratprognose beispielhaft diskutiert, wie Daten über geeignete Schnittstellen aus bestehenden Gewässergütedatenbanken zugänglich gemacht und Messwerte verschiedener Quellsystemen kombiniert werden können, um damit ein Prognosemodell zu trainieren. Es wird gezeigt, dass auf Basis des zweiten Teils der SensorThings API (Tasking Core), eine flexible Integration von KI-Algorithmen möglich ist. Die Prognose soll im Rahmen des NiMo-Projektes in ein Decision Support System für Entscheidungsträger in Politik und Industrie einfließen.

Um den Anforderungen der verteilten Zuständigkeiten Rechnung zu tragen wird dabei ein Vorgehensmodell für die Integration von KI-Algorithmen angewendet.

3 Herausforderungen bei der Integration von KI-Algorithmen in Umweltinformationssysteme

Bei der Integration von KI-Algorithmen in Umweltinformationssysteme treten einige Herausforderungen auf, die im Folgen zusammengefasst werden sollen um den Kontext und die Zielsetzung diese Beitrags zu motivieren.

Die erste Schwierigkeit liegt darin, dass sich der Lebenszyklus von etablierten Umweltinformationssystemen und KI-Anwendungen unterscheidet. Erstere sind üblicherweise über Jahre, bzw. Jahrzehnte umgesetzt und erweitert worden. Weiterentwicklungen folgen klassischen Softwareentwicklungsprozessen. Das Einbringen und Evaluieren neuer Komponenten und Algorithmen ist daher mit einem hohen Aufwand verbunden. Dieses Vorgehen steht im Gegensatz zur agil geprägten Entwicklung von KI-Anwendungen. Aufgrund der Unsicherheit bezüglich Datenlage und Eignung verschiedener Lösungsansätze kann eine KI-Anwendung, insbesondere die eingesetzte KI-Methode nicht vollständig im Vorfeld geplant und umgesetzt werden. Diese kann nur in einem iterativen Proof-of-Concept Prozess, bestehen aus prototypischer Umsetzung und Evaluierung anhand vorhandener Daten (beispielsweise beschreiben in [JP20]) entwickelt werden. Um den produktiven Einsatz eines bestehenden Umweltinformationssystems nicht zu beeinträchtigen, ist eine passende Test- und Evaluationsumgebung zu schaffen.

Für die Entwicklung und Evaluierung von KI-Methoden müssen geeignete Daten zur Verfügung stehen. Hierfür können synthetische Werte generiert werden, besser geeignet sind jedoch reale Daten. Diese stehen in bestehenden Umweltinformationssystemen über teils proprietäre Schnittstellen zur Verfügung. Aufgrund der Unsicherheit ist im Vorfeld häufig nicht bekannt welche Datenquellen benötigt werden. Daher lassen sich verschiedene Datenquellen mit unterschiedlichen Zugriffsprotokollen nur mit erheblichem Aufwand anbinden. Um flexibel den Einfluss verschiedener Quellen evaluieren zu können, müssen diese einfach zugreifbar zur Verfügung stehen. Neben einer einheitlichen Zugriffsschnittstelle wird ein gemeinsames Datenmodell mit definierter Semantik benötigt.

Zusätzlich zu den Herausforderungen bei der Bereitstellung der Daten, muss die Integration und der Betrieb von KI-Algorithmen betrachtet werden. Das beschriebene agile Vorgehen bei der Entwicklung führt dazu, dass diese häufig in Form von Jupiter Notebooks[4] oder Python-Scripten bereitgestellt werden. Diese, aus Softwaretechnik-Sicht *ad hoc*-Entwicklung führt zu fehlender Wiederverwendbarkeit und Nachvollziehbarkeit. Hat sich eine Proof-of-Concept Implementierung bewährt, wird ein transparentes Vorgehen zur Überführung in den in produktiven Betrieb benötigt. Erschwerend kommt hinzu, dass im Bereich der Datenauswertung und bei der Entwicklung von KI-Algorithmen eine Vielzahl unterschiedlicher Programmiersprachen, Frameworks und Bibliotheken eingesetzt werden.

Im Rahmen des NiMo-Projektes werden daher folgende Ziele zur Integration von KI-Algorithmen in Umweltinformationssysteme verfolgt: Erstens, eine einheitliche Bereitstel-

[4] https://jupyter.org/

lung unterschiedlicher Datenquellen. Zweitens eine Plattform zur einfachen Integration von KI-Algorithmen. Diese soll es erlauben bestehende Entwicklungen ohne größere Anpassung integrieren zu können. Ein Anwender soll außerdem den KI-Algorithmus ohne tiefergehendes System- und Administrationswissen zur Ausführung bringen können. Hierzu soll sowohl über die genaue Funktion und technische Umsetzung der KI-Methode, als auch von der Infrastruktur zu Ausführung abstrahiert werden. Die Berechnung soll durch eine einfache Auswahl des KI-Algorithmus und der notwendigen Konfigurationsparamerter angestoßen werden können.

4 Related Work

Das Umweltbundesamt hat mit der Studie „Künstliche Intelligenz im Umweltbereich“ Zukunftsperspektiven für den Einsatz von KI in der Umweltdomäne untersucht [TJ19]. Haupttreiber der Entwicklung von KI-Awendungen sind wirtschaftliche Akteure. Es wird herausgestellt, dass intelligente Systeme komplexe Zusammenhänge deuten und Muster erkennen können und damit als flexible Warnsysteme für den Umweltschutz dienen können. Ein Anwendungsbeispiel für die Herausforderung des Einhaltens von Grenzwerten ist die im Anwendungsbeispiel gewählte Nitratprognose. Die Ergebnisse können nicht nur Einflüsse auf die Prozesse der Wasserversorger, sondern auch Einfluss auf politische Entscheidungen haben. Der Artikel „Übertragung des Vorgehensmodells KI auf Umweltinformationssysteme“ beleuchtet ein mögliches Vorgehensmodell um diesen Prozess im behördlichen Umfeld zu unterstützen [HP21]. Die technische Umsetzung bleibt aber offen. Ein open source Ansatz für die Modellierung von Schadstoff wird mit MoRE (Modelling of Regionalized Emissions) verfolgt [Fu17]. MoRE koppelt einen Rechenkern mit einer Datenbank und stellt die Ergebnisse über einen Webserver tabellarisch und in From von Diagrammen zur Verfügung. Eine offene, standardisierte Schnittstelle zur Datenabfrage stellt das System nicht zur Verfügung. Ein gängiger Standard zur Bereitstellung solcher Daten ist der OGC Sensor Observation Service (SOS) [BSE12]. Im Gegensatz zur SensorThings API [LHK16] setzt der SOS auf Prinzipien einer Service-oriented architecture [SW04]. Dadurch steigt die Komplexität und der Aufwand SOS einzusetzen. Eine offener Standard zu Integration von Auswertungen ist der OGC Web Processing Service (WPS) [Sc07]. Beilspielsweise wird dieser von Wiemann et al. zur Modellintegration [Wi18] aufgegriffen. Ähnlich wie der Sensor Observtion Serivce ist auch WPS komplex in der Verwendung und hat daher in diesem Kontext keine breitere Verwendung gefunden.

Für das Einbringen und Ausführen von Verarbeitungsschritten zur Auswertung von heterogenen Datenquellen gibt es insbesondere aus dem Bereich der BigData-Verarbeitung unterschiedliche existierende Plattformen [Ma17]. Hierzu zählt beispielsweise Apache Hadoop[5], das auf MapReduce [DG08] als Programmiermodell setzt. Als Alternative existiert mit Spark ein weiteres Werkzeug für die Verarbeitung großer Datenmengen [Za16]. Im Gegensatz zu Hadoop, das auf MapReduce ausgerichtet ist, unterstützt Spark neben der

[5] https://hadoop.apache.org/

Stream- und Graphverarbeitung auch SQL-Abfragen sowie Machine Learning Modelle. Für die Umsetzung von Verarbeitungsschritten stehen sowohl bei Hadoop als auch bei Spark geeignete Schnittstellen zur Integration bereit. Dadurch sind sowohl das Programmiermodell als auch die unterstützten Programmiersprachen vorgegeben. Eine freie Technologiewahl ist nicht möglich. Daher lassen sich bestehender Implementierungen häufig nur mit großem Aufwand integrieren.

Speziell für den Einsatz von KI-Methoden steht mit Kubeflow [Bi19] eine Plattform für das Entwickeln, Trainieren und Ausführen von Machine Learning Anwendungen zur Verfügung. Kubeflow unterstützt KI-Algorithmen von der Entwicklung und dem Trainieren bis hin zur skalierbaren Ausführung. Dadurch ist Kubeflow ein mächtiges Werkzeug für den kompletten Lebenszyklus einer KI-Anwendung. Dies setzt jedoch eine entsprechende Einarbeitung und Aufwand bei der Integration bestehender Entwicklungen voraus.

5 Integration über SensorThings API

Grundlage für die Daten- und KI-Integration in diesem Beitrag bildet der offene und leichtgewichtige Geodatenstandard *OGC SensorThings API*. Dabei wird für die Bereitstellung der messreihenbasierten Daten auf den ersten Teil (*Part 1: Sensing*) des Standards gesetzt. Zur Einbindung der KI-Algorithmen, deren Konfiguration und Ausführung wird auf Teil zwei (*Part 2: Tasking Core*) zurückgegriffen.

5.1 SensorThings API

Die OGC SensorThings API ist ein Standard zur Verwaltung von Sensormess und -metadaten. Im Gegensatz zu dem bereits genannten Sensor Observation Service baut die SensorThings API auf moderne Konzepte und Technologien auf. Hierzu zählt eine JSON-basierte Datenrepräsentation, eine REST-Schnittstelle [Fi00] zur Datenabfrage und Manipulation sowie eine auf OData [PHZ14] aufbauende Abfrage- und Filtersprache. Dies ermöglicht eine einfache und technologieunabhängige Bereitstellung von Sensordaten. Dadurch hat die Verbreitung der SensorThings API für den Zugriff auf Sensordaten in den letzten Jahren zunehmen an Bedeutung gewonnen.

Die OGC SensorThing API besteht aus den folgenden Komponenten: 1. einem Datenmodell das vorhandene Entitäten und deren Beziehungen definiert 2. einer HTTP basierten Schnittstelle, die das Abfragen, Bearbeiten und Löschen von Daten erlaubt 3. mehreren Erweiterungen, wie beispielsweise eine MQTT-Schnittstelle die die Umsetzung ereignisbasierter Anwendungsfälle erlaubt.

Neben dem *OGC SensorThings API Part 1: Sensing*, existiert ein weiterer Standard, die *SensorThings API Part 2:* **Tasking Core** [LK19]. Dieser Teil des Standards verfügt über die interessante Möglichkeit mittels der leichtgewichtigen Schnittstelle nicht nur Daten

abzufragen, sondern auch Berechnungsprozesse anzustoßen. Das Datenmodell des ersten Teils wird um weitere Entitäten (Abbildung 1) ergänzt. Dadurch können mit Hilfe der SensorThings API nicht nur Sensor Mess- und Metadaten, sondern auch Aktuatoren modelliert werden. Dies bietet die Grundlage für die in diesem Beitrag betrachtete KI-Integration.

Ein *Actuator* ist häufig ein physikalisches Gerät, das gesteuert werden kann. Die Fähigkeiten des Aktors werden als *TaskingCapability* dargestellt. Diese wiederum sind einem *Thing* zugeordnet, was den Übergang in das Datenmodell des ersten Teils darstellt. Während eine *TaskingCapability* die Fähigkeiten beschreibt, werden Aktionen des Aktors mittels eines *Task*s umgesetzt. Akzeptierte Konfigurationsparameter einer *TaskingCapability* werden über das Attribut *taskingParameters* definiert. Die konkreten Werte zur Ausführung werden im *Task* über das gleichnamige Attribut abgelegt. Betrachtet man einen Fensteröffner als *Thing*, so besitzt dieser einen Motor (*Actuator*), der das Fenster zu einem gewissen Prozentsatz öffnen kann (*TaskingCapability*). Soll das Fenster nun zu 50% geöffnet werden, wird eine *Task*-Instanz erzeugt, die als *taskingParameter* einen Wert *öffnung* = 50% enthält. An dieser Stelle sei angemerkt, dass der Standard keine Aussage über die Darstellung des Ausführungszustands macht.

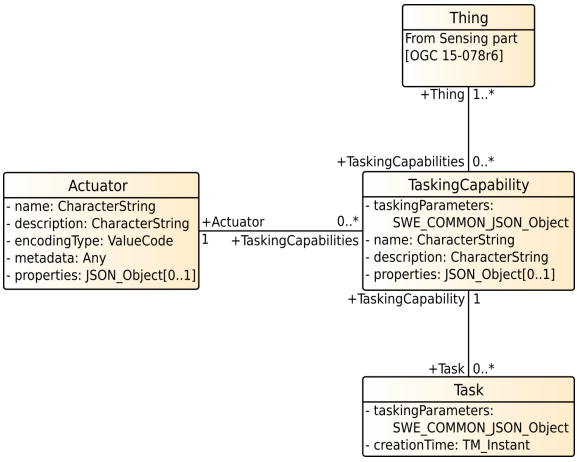

Abb. 1: Datenmodell der OGC SensorThings API Part 2: Tasking Core

Ein Aktor muss nicht zwingend ein physikalisches Objekt repräsentieren. Ebenfalls können reine Softwarekomponenten (im Kontext dieses Artikels *virtueller Aktor* genannt) betrachtet werden. Aufrufe an die Softwarekomponente können somit als *Task* und Aufrufparameter als *taskingParameter* dargestellt werden. Betrachtet man mit diesem Hintergrund einen KI-Algorithmus, so kann dieser als *virtueller Aktor* aufgefasst und das Anstoßen einer Berechnung als *Task* umgesetzt werden.

5.2 Bereitstellung der Daten über die SensorThings API

Mit Hilfe der SensorThings API Part 1 wird im Projekt NiMo die Bereitstellung von dynamischen Zeitreihendaten realisiert. Für die Umsetzung des Anwedungsbeispiels der Nitratprognosse sind Gütemesswerte von Gewässern notwendig. Diese stehen in einer internen Umweltdatenbank zur Verfügung. In einem ersten Schritt wurde das vorhandene Datenmodell auf die SensorThings API abgebildet. Ein *Thing* repräsentiert eine Messstelle aus einer Datenbank mit Grundwasserqualitätsdaten.Die Koordinaten des Messortes wurden auf die *Location* abgebildet. Die Kombination des gemessenen Parameters (z.B. Nitrat) mit dem Medium in dem dieser gemessen wurde (z.B. im Anwendungsfall Wasser) entspricht der *ObservedProperty*. Auf den *Datastream* wurden die allgemeinen Attribute einer Zeitreihe eines Messortes abgebildet. Für die Ermittlung der Messwerte werden an klassischen Messstellen Wasserproben genommen, welche im Labor analysiert werden. Die Informationen über die Probenahme wurden im *FeatureOfInterest* abgebildet. Die Messwerte werden durch die *Observations* repräsentiert.

Da eine Anpassung am Quellsystem nicht möglich ist, um dies beispielsweise um eine SensorThings API Schnittstelle zu erweitern, wurden die vorhandenen Daten in eine Instanz von FROST®[6] kopiert. Dieser ist eine Open Source Referenzimplementierung der SensorThings API und bildet im Rahmen von NiMo die Grundlage zur Daten und Algorithmenintegration.

5.3 PERMA: Integration von Verarbeitungsschritten

Im Abschnitt 1 wurde aufgezeigt, dass sich die SensorThings API Part 2: Tasking Core neben der Modellierung von physikalischen Aktuatoren auch dazu eignet Metadaten von Verarbeitungsschritten, bzw. KI-Algorithmen zu modellieren, deren Ausführung durch *Tasks* darzustellen und die Ausführungsparameter über *TaskingParameters*

In der praktischen Umsetzung ergeben sich an dieser Stelle, wie bereits in Abschnitt 3 motiviert, weitere Fragestellungen. *Virtuelle Aktoren*, bzw. KI-Algorithmen benötigen ein System oder eine Laufzeitumgebung, auf der sie ausgeführt werden können. Da KI-Algorithmen häufig von Datenwissenschaftlern oder Fachexperten umgesetzt werden, besitzen diese in der Regel nicht das nötige Wissen um die Implementierung (bspw. per Secure Shell (ssh)) auf einem Server zur Ausführung zu bringen.

Um dieser Herausforderung zu begegnet, wird im Rahmen von NiMo eine Software-Komponente namens PERMA entwickelt und eingesetzt. Mit Hilfe von PERMA können KI-Algorithmen ohne tieferes Wissen über die eigentliche Infrastruktur, bzw. Laufzeitumgebung zur Ausführung gebracht werden. Im Rahmen des Projektes werden hierfür zwei Technologien unterstützt. Erstens können KI-Algorithmen in der Programmiersprache Java

[6] `https://github.com/FraunhoferIOSB/FROST-Server`

umgesetzt werden. Dazu muss lediglich ein einfaches Interface mit zwei Methoden (Abfrage der bereitgestellten *TaskingCapabitlity* und Ausführen der Berechnung) implementiert werden. Anschließend kann dies als JAR-Artefakt gepackt, in PERMA hochgeladen und zur Ausführung gebracht werden.

Je nach Anwendungsfall werden KI-Algorithmen mit unterschiedlichen Technologien umgesetzt, wofür eine geeignete Abstraktionsebene notwendig ist. Hierfür setzt PERMA auf Container-Technologien. Dies erlauben es die KI-Algorithmus zusammen mit abhängigen Bibliotheken, Frameworks und Tools in ein austauschbares und ausführbares Artefakt zu packen. Neben einer Java-Laufzeitumgebung unterstützt daher PERMA als zweite Technologie den Einsatz von Containern. Dies ermöglicht es KI-Algorithmus in nahezu jeder Programmiersprache zu implementieren und zu integrieren.

Klassische Container-Plattformen wie Kubernetes oder openShift[7] sind aufgrund ihrer vielfältigen Funktionen mächtig, setzten jedoch einen großen Einarbeitungsaufwand voraus. Ähnliches gilt für Machine Learning Frameworks wie Kubeflow. PERMA bietet hier eine zusätzliche Abstraktionsebene um diese Komplexität zu verstecken. Entwickler der KI-Algorithmen müssen lediglich ein Container Image erstellen, das Eingabeparameter als *TaskingParameter* definiert und die Werte aus den Umgebungsvariablen auslesen kann. Für das Ausführen der Berechnung kann, ohne weitere Kenntnisse, über die PERMA Weboberfläche angestoßen werden. Durch das Anlegen eines *Tasks* kann dies ebenfalls durch ein externes System erfolgen. Zur Ausführung der Container setzt PERMA im Hintergrund auf Kubernetes. Eine Erweiterung, dass lediglich Docker[8] zur Verfügung steht, ist in Planung.

6 Anwendungsbeispiel Nitratmonitoring

Im Rahmen des Forschungsprojektes NiMo wird die Integration eines KI-Algorithmus mit Hilfe der SensorThings API anhand einer flächenhaften Prognose von Nitratdaten demonstriert. Hierzu wird das *Vorgehensmodell Machine Learning for Production (ML4P)* [JP20] aus dem Kontext der industriellen Produktion auf ein Umweltinformationssystem übertragen [HP21]. Die einzelnen Phasen des methodischen Vorgehens sind: 1. *Zieldefinition und Lösungsansatz*, 2. *Proof of Concept*, 3. *Systemspezifikation*, 4 *Umsetzung und Inbetriebnahme*, 5. *Übergabe* und 6. *Betrieb*.

Im ersten Schritt (*Zieldefinition und Lösungsansatz*) wurde im Rahmen des Projektes ein Anwendungsfall definiert, mit dem die Integration eines KI-Algorithmus in ein Umweltinformationssystem demonstriert werden kann. Hierfür wurde die flächenhafte Prognose von Nitrat im Grundwasser mit Hilfe von an Messstellen gemessenen Nitratwerten ausgewählt.

[7] `https://www.openshift.com/`
[8] `https://www.docker.com/`

Im Rahmen der *Proof of Concept* Phase wurde ein Lösungsansatz prototypisch umgesetzt und die Machbarkeit überprüft. Die in die Berechnung einfließenden Datenquellen standen zu diesem Zeitpunkt noch nicht endgültig fest. Daher wurden potentiell relevante Datenquellen bereitgestellt. Dies sind einerseits räumliche Daten, bzw. Rasterdaten, welche statisch sind und sich nicht regelmäßig ändern, beispielsweise Informationen über Flächennutzung, Stickstoffüberschuss und -hintergrunddeposition, Bodenbeschaffenheit, Sickerwasserrate und Wasserschutzgebiete. Andererseits wurden dynamische Zeitreihendaten beispielsweise die der Nitratmessstellen aus öffentlich verfügbaren Grundwasserqualitätsdaten betrachtet. Eine direkte Integration dieser Daten für das Vorhersagemodell ist aufgrund der uneinheitlichen und nicht standardisierten Schnittstelle nicht praktikabel. Aus diesem Grund wurden, wie in Abschnitt 5.2 beschrieben, Gewässerqualitätsdaten aus vorhandenen Quellsystemen auf das Datenmodell der SensorThings API abgebildet und in eine FROST-Instanz importiert. Damit ergeben sich einerseits Synergieeffekte für den Datenquellbezug bei der Entwicklung weitere KI-Algorithmen mit ähnlichen Daten und die Möglichkeit die verfügbaren Daten länderübergreifend in einem System zu nutzen.

Bei der prototypischen KI-Methode handelt es sich um einen Algorithmus zur räumlichen Vorhersage der Nitratkonzentration im Grundwasser. Die Grundlage stellt ein Random Forest dar, der in der Programmiersprache Python geschrieben wurde und ein Regressionsproblem löst. Für dieses Problem stehen dem Algorithmus die Nitratwerte der Grundwassermessstellen als Zieldaten sowie die genannten Rasterdaten als Eingangsdaten zur Verfügung. In einem ersten Schritt werden aus dem Gesamtdatensatz Trainings-Daten erzeugt. Dies geschieht durch die räumliche Abfrage der Rasterwerte an jeder der Nitratmessstellen. So entsteht eine 2D-Matrix, in der jede Zeile eine Nitratkonzentration sowie die zugehörigen Eingangsdaten enthält. Zusätzlich zu dieser Trainings-Matrix wird noch eine 3D-Matrix der Größe [Länge*Breite*Anzahl Eingangsdaten] erstellt, die als Grundlage der räumlichen Vorhersage dient. Anschließend wird der Random Forest mit vorher optimierten Hyperparametern erstellt und mit den Trainings-Daten trainiert, um die Zusammenhänge zwischen Eingangsdaten und Nitratkonzentrationen zu lernen. Nachdem das Modell trainiert wurde, erfolgt die Vorhersage der Nitratkonzentration über die Fläche. Hierfür iteriert der Algorithmus über die 3D-Matrix und berechnet an jeder Stelle einen Wert, sodass als Ergebnis eine Matrix mit räumlich vorhergesagten Nitratkonzentrationen entsteht. Fehlen an einer Stelle Informationen zu Eingangsdaten, wird an dieser Stelle kein Wert vorhergesagt, sondern diese Zelle übersprungen. Aus der berechneten Matrix mit den räumlich vorhergesagten Nitratkonzentrationen wird dann wieder eine Raster-Datei mit räumlichem Bezug erzeugt.

Die Entwicklung des KI-Algorithmus wurde in der Programmiersprache Python umgesetzt und vorhandene Entwicklungswerkzeuge eingesetzt. Die Funktionsfähigkeit des *Proof of Concept*s konnte somit ohne zusätzlichen Einarbeitungsaufwand gezeigt werden.

Im Rahmen der *Systemspezifikation* wurden die genannten Herausforderungen bei der Überführung der prototypischen Implementierung vom Entwicklersystem in eine produktive Laufzeitumgebung betrachtet, damit die Prognose weiteren Anwendern zur Verfügung gestellt werden kann. Die Grundstruktur der Systemspezifikation ist durch den Einsatz von

PERMA, bzw. der SensorThings API bereits vorgegeben. Sodass in diesem Schritt lediglich der Prognose-Algorithmus und dessen Parameter [SL18] durch eine *TaskingCapability* beschrieben werden mussten.

Zur *Umsetzung und Inbetriebnahme* wurde für die vorhandene Implementierung, zusammen mit der *TaskingCapability* aus der vorangegangenen Phase ein Container-Image erzeugt. Dies ist beispielsweise mit Hilfe von Docker[9] ohne großen Aufwand möglich. Damit sind die Voraussetzungen erfüllt, um den KI-Algorithmus mit Hilfe der in Abschnitt 5.3 beschriebenen Komponente PERMA zur Ausführung zu bringen. In einem ersten Schritt muss dazu die Nitratprognose, unter Angabe des Containerimages, in PERMA registriert werden. Danach kann eine Ausführung, ohne größere Kenntnisse und technisches Wissen, über die PERMA-Weboberfläche angestoßen werden. Die abschließenden Phasen *Übergabe* und *Betrieb* wurden im Projekt noch nicht durchlaufen. PERMA selbst setzt auf Kubernetes[10] als Containerlaufzeitumgebung. Sodass in diesen Phasen eine entsprechende Umgebung in der Zielinfrastruktur bereitgestellt werden muss.

Basis für die Datenbereitstellung und KI-Integration ist die SensorThings API. Aufgrund dieser standardisierten und offenen Schnittstelle kann dies problemlos in einen größeren Systemkontext integriert werden. Im Rahmen des NiMo-Projektes soll die Vorhersage der Nitratbelastung beispielsweise in ein Entscheidungsunterstützungssystem einfließen.

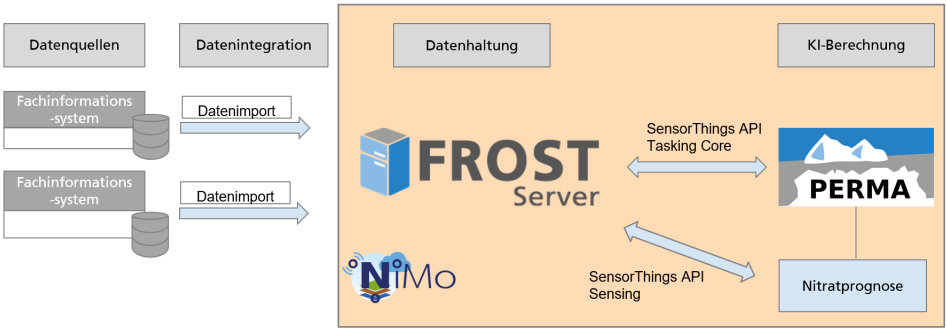

Abb. 2: NiMo Architektur

Die aus dem ersten Anwendungsfall abgeleitete NiMo-Architektur ist in Abbildung Abbildung 2 verfügbar. Es hat sich gezeigt, dass alle bisher betrachteten Datenquellen mit Zeitreihendaten problemlos und ohne größeren Aufwand in das SensorThings API Datenmodell überführt werden können. Aufgrund der unterschiedlichen Eigenschaften und Datenrepräsentationen ist für jedes Quellsystem ein individueller Datenimport notwendig. Aktuell laufen weitere Arbeiten um diesen Schritt durch geeignete Import-Werkzeuge zu vereinfachen. Durch die standardisierte Bereitstellung der Daten konnte die Entwicklung der Nitratprognose deutlich vereinfacht werden. Zusätzlich konnte in der *Proof of Concept*-Phase mit geringem Aufwand der Einfluss unterschiedlicher Eingabedaten auf das

[9] https://www.docker.com
[10] https://kubernetes.io/

Vorhersagemodell evaluiert werden. Im Vergleich zur Datenbereistellung über das lokale Dateisystem führt der Datenzugriff über die Sensorthings API, bzw. das HTTP-Protokoll zu Performanz-Einschränkungen. In der Praxis wurden jedoch keine erheblichen Einbußen beobachtet, die dieses Zugriffsmuster unpraktikabel oder ein Zwischenspeichern der Daten erforderlich machen. Vielmehr wurde dadurch die Flexibilität bei der Entwicklung der Prognose erhöht. Durch den Einsatz von PERMA ist es möglich die Nitratprognose ohne großen Aufwand vom Entwicklungssystem in eine produktive Umgebung zu überführen und somit anderen Beteiligten schnell zur Verfügung zu stellen.

7 Zusammenfassung und Ausblick

In diesem Artikel wurde diskutiert, wie KI-Algorithmen über standardisierte offenen Schnittstellen in bestehende Systeme integriert werden können. Beschrieben wurde ein Ansatz der dies mittels der SensorThings API umsetzt. Anhand des praktischen Anwendungsfalls zeigt sich, dass die SensorThings API mit ihrem zeitreihenbasierten Datenmodell für Integration der Gewässerqualitätsdaten gut geeignet ist. Für die Realisierung des endgültigen Systems werden außerdem Rasterdaten benötigt. Wie diese über weitere offene Schnittstellen behandelt werden können wird im weiteren Verlauf des Projektes NiMo untersucht. Durch den Einsatz von PERMA und der SensorThings API: Part 2 konnte die implementierte Nitratprognose erfolgreich und ohne großen Mehraufwand in einer Zielumgebung zur Ausführung gebracht werden. Dadurch kann eine vom Umweltinformationssystem unabhängige Entwicklung der KI-Methoden gewährleisten und eine Erprobung dieser ermöglicht werden. Im weiteren Verlauf des Projektes wird dieses Vorgehen auf weitere Anwendungsbereiche angewendet und anhand von Performance-Tests der Einfluss des Zugriffs auf die SensorThings API weiter untersucht.

8 Acknowledgement

Diese Arbeit wurde im Rahmen des vom BMU geförderten Projekts *„NiMo 4.0“* [Li21] (Förderkennzeichen 67KI2048) erstellt.

Literaturverzeichnis

[Bi19] Bisong, Ekaba: Kubeflow and kubeflow pipelines. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform, S. 671–685. Springer, 2019.

[BSE12] Bröring, Arne; Stasch, Christoph; Echterhoff, Johannes: OGC Sensor Observation Service Interface Standard, Version 2.0. 2012.

[DG08] Dean, Jeffrey; Ghemawat, Sanjay: MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1):107–113, 2008.

[Fi00] Fielding, Roy Thomas: Architectural styles and the design of network-based software architectures. University of California, Irvine, 2000.

[Fu17] Fuchs, Stephan; Weber, Tatyana; Wander, Ramona; Toshovski, Snezhina; Kittlaus, Steffen; Reid, Lucas; Bach, M; Klement, L; Hillenbrand, T; Tettenborn, F: Effizienz von Maßnahmen zur Reduktion von Stoffeinträgen. Endbericht. UBA-Texte, 5:2017, 2017.

[HP21] Hilbring, Désirée; Pfrommer, Julius: Übertragung eines Vorgehensmodells zur KI-Integration von der Industrie auf Umweltinformationssysteme. INFORMATIK 2020, 2021.

[JP20] Julius Pfrommer, Jürgen Beyerer, Marlene Eisenträger Christian Frey Andreas Herzog Ali Moghiseh Lukas Morand Henrike Stephani Anke Stoll Lars Wessels: ML4P – Vorgehensmodell Machine Learning for Production. White paper, Fraunhofer, 10 2020.

[LHK16] Liang, Steve; Huang, Chih-Yuan; Khalafbeigi, Tania: OGC SensorThings API Part 1: Sensing, Version 1.0. 2016.

[Li21] Liesch, Tanja; Bruns, Julian; Abecker, Andreas; Hilbring, Désirée; Karimanzira, Divas; Martin, Tobias; Wagner, Martin; Wunsch, Andreas; Fischer, Thilo: Nitrat-Monitoring 4.0 – Intelligente Systeme zur nachhaltigen Reduzierung von Nitrat im Grundwasser. In (Reussner, Ralf H.; Koziolek, Anne; Heinrich, Robert, Hrsg.): INFORMATIK 2020. Gesellschaft für Informatik, Bonn, S. 1069–1079, 2021.

[LK19] Liang, Steve; Khalafbeigi, Tania: OGC SensorThings API Part 2–Tasking Core, Version 1.0. 2019.

[Ma17] Marjani, Mohsen; Nasaruddin, Fariza; Gani, Abdullah; Karim, Ahmad; Hashem, Ibrahim Abaker Targio; Siddiqa, Aisha; Yaqoob, Ibrar: Big IoT data analytics: architecture, opportunities, and open research challenges. ieee access, 5:5247–5261, 2017.

[PHZ14] Pizzo, Michael; Handl, Ralf; Zurmuehl, Martin: OData Version 4.0 Part 1: Protocol Plus Errata 02. 2014.

[Sc07] Schut, Peter: OpenGIS® Web Processing Service. 2007.

[SL18] Steve Liang, Tania Khalafbeigi, Kan Luo: OGC SensorThings API Tasking CoreDiscussion Paper. Discussion paper, OGC, 2018.

[SW04] Sprott, David; Wilkes, Lawrence: Understanding service-oriented architecture. The Architecture Journal, 1(1):10–17, 2004.

[TJ19] Tobias Jetzke, Stephan Richter, Jan-Peter Ferdinand und Samer Schaat: , Künstliche Intelligenz im Umweltbereich. https://www.umweltbundesamt.de/sites/default/files/medien/1410/publikationen/2019-06-04_texte_56-2019_uba_ki_fin.pdf, 2019.

[Wi18] Wiemann, Stefan; Al Janabi, Firas; Eltner, Anette; Krüger, Robert; Luong, T; Sardemann, Hannes; Singer, Thomas; Spieler, Diana; Kronenberg, Rico: Entwicklung eines Informationssystems zur Analyse und Vorhersage hydro-meteorologischer Extremereignisse in mittleren und kleinen Einzugsgebieten. In: Forum Hydrol. Wasserbewirtsch. Jgg. 39, S. 357–367, 2018.

[Za16] Zaharia, Matei; Xin, Reynold S; Wendell, Patrick; Das, Tathagata; Armbrust, Michael; Dave, Ankur; Meng, Xiangrui; Rosen, Josh; Venkataraman, Shivaram; Franklin, Michael J et al.: Apache spark: a unified engine for big data processing. Communications of the ACM, 59(11):56–65, 2016.