Integrating Graph and Large Language Models with AOP-Wiki for Contextual and Semantic Parsing of Adverse Outcome Pathway Information

Saurav Kumar[1], Deepika Deepika[1], Luke T Slater[3], Vikas Kumar[1,2]*

[1]IISPV, Hospital Universitari Sant Joan de Reus, Universitat Rovira I Virgili, Reus, Spain
[2]German Federal Institute for Risk Assessment (BfR), Berlin, Germany
[3] Center for Environmental Research and Justice, University of Birmingham, United Kingdom

Abstract

Background: Adverse Outcome Pathways (AOPs) provide a basis for non-animal testing, by outlining and encoding the cascade of molecular and cellular events initiated upon stressor exposure, leading to adverse effects. AOPs are therefore considered an integral part of the New Approach Methodology (NAMs) to human health risk assessment. Over the last few years, the scientific community has shown immense interest in developing AOPs with crowdsourcing, with the results of those efforts being archived in the AOP wiki: a centralized repository coordinated by the OECD, which hosts nearly 512 AOPs. However, the AOP-wiki platform currently lacks a versatile querying system, which would enable developers to integrate and explore the AOP network data in a manner that would permit their practical use in analysis. In the context of the recent community attention on developing more and more expressive AOPs, this limitation is particularly relevant.

Implementation: This work proposes to expose the full potential of the AOP wiki archive by adapting its data into a label property graph (LPG) database. In addition, the solution provides both database-specific query and natural language query interfaces to support retrieval and integration of the graph data, including the ability to chain stepwise queries. To evaluate the platform, a case study is presented, with three levels of use case scenario (simple, moderate, and complex query). This tool is freely available on GitHub for wider community usability and further enhancement.

Conclusions: The presented query engine increases the potential of AOP exploration, by reducing the time, human resources, and technical ability needed to make practical use of the data they contain. Also, its dynamic prompt generation capability, will lead to precise query generation as the interaction increases. We therefore anticipate that the approach will be integral in future realisation of new approach methodologies in chemical risk assessment.

Keywords: Adverse outcome pathway, large language model, Graph database, Risk assessment, Artificial intelligence, data integration, information retrieval, information extraction

Introduction

The concept of Adverse Outcome Pathway (AOP) was defined in 2010 by Ankely et al. to streamline the idea of Next Generation Risk Assessment (NGRA)[1]. Ankely et al. described AOPs as an organizing framework to facilitate the representation of cascade events initiated by the sufficient stressor's perturbation at the molecular level. Events in AOPs are broadly categorized into three sub-categories i.e., Molecular Initiating Event (MIE), Key Event (KE), and Adverse Outcome (AO), with the respective biological organization level based on their occurrence to effectively apply for risk assessment[1,2]. AOPs broadly capture the mechanistic knowledge of events in a well-connected sequential format, which helps in the strategic planning and development of new approach methodologies (NAMs) such as in vitro tests, targeted assays, and Integrated Approaches to Testing and Assessment (IATA)[1–3]. The NAMs aim to fill the gaps in decision-making in chemical risk assessment.

AOP developments are backed by guidelines and principles mentioned in the AOP Developers' Handbook, which is revised regularly to reflect feasible principles for improving the AOPs [4]. As per the mentioned guidelines, AOPs developed over the years are stored in AOP Knowledgebase (AOP-KB). AOP-KB stores machine-readable textual information of AOPs in XML (Extensible Markup Language) formatted data. On top of AOP-KB, there are two other services i.e., AOP Portal and AOP Wiki, that acts as query interface. The AOP portal enables the search of AOP and key events with keywords, it also provides information about AOP endorsement. Whereas AOP-wiki is hosted as a central repository for AOPs developed as part of the OECD AOP development Programme. AOP-Wiki provides a platform to crowd-source and organize available knowledge as well as provides read/write access to AOP-KB following the OECD EAGMST guidelines. There are various third-party tools developed to enrich and support the AOP development which are assembled in the AOP-wiki and AOP-KB platform. But none of the tool till now has implmented AOP data in its native format i.e., graph.

The ubiquity of graphs in data representation cannot be overstated. Any form of data can be reimagined as graph data whether it is tabular, relational, or even unstructured text [5]. In mathematical terms graphs are represented with notation $G = (V, E),$ where V is the set of nodes and E is the set of edges, where each edge contains a pair of nodes (u, v) representing a connection between node u and node v [5]. The implicit data structure of AOPs is a graph as it represents the cascade of events in the form of nodes (KE) connected by edges (KER). Building upon this implicit graph structure, initial efforts have been invested in modeling AOPs using RDF (Resource Description Framework) triples. RDF is one of the approaches for representing and modeling data in graph format. Notable examples include the work done by Ives et al. on systematically breaking down the key event term of AOP into three sub-terms i.e., process, object, and action. Each of the sub-terms were linked with respective ontologies and assigned a unique identifier [6].

Similarly, Martens et al. converts AOP data into Resource Description Format (RDF) by linking terms such as AOP, key event, and key event relationship with persistent and resolvable identifiers, which allow interoperability with other resources [7]. Introduction of RDF presents a paradigm shift in data modelling by encapsulating data as subject-predict-object triples, enabling a structured and semantically rich framework for describing AOP[3,7,8]. The adoption of RDF aligns with the principles of Data FAIRification, which advocate for data that is Findable, Interoperable and Reusable [9]. RDF modelled data is queried using SPARQL (SPARQL Protocol and RDF Query Language), which allows searching, filtering, and extracting of the desired information. RDF conversion makes AOPs query able up to a certain extent, but the complex and lengthy query of SPARQL makes it hard for the non-technical user to implement it. Also, querying the resources in SPARQL required the exact identifier (Uniform Resource Identifiers) of the nodes, which is troublesome to remember. The response of SPARQL queries is in tabular format in AOP-WIKI RDF tool which is counterintuitive as per the graph nature of AOPs[3,7] The tabular response gives a hard time to the user to understand, how various AOP elements are interconnected. It does not effectively illustrate how individual biological processes are orchestrated by multiple events within the complex network structure. Furthermore, SPARQL lacks the flexibility for deep-data traversals, path analysis and has logarithmic cost of graph transversal [10–12].

In this article, we unveil the capacity of a Labeled Property Graph (LPG) data modelling paradigm to serve as a natural data structure for Adverse Outcome Pathways (AOP). In LPG, data is organized into nodes and relationships in contrast with RDF-triples which consist of subject-predicate-object. We demonstrate how an LPG-adapted graph database empowers the risk assessors, and modelers to easily capture the multifaceted relationship that underlies within AOPs, while efficiently exploiting the wealth of unstructured information associated with each AOPs. Crafting queries either in RDF SPARQL

or graph for data retrieval is a challenge, particularly for non-technical users. To alleviate this complexity, a pioneering step has been taken to integrate a powerful Large Language Model (LLM). The LLM model with its ability to seamlessly generate queries in response to natural language queries by users, bridges the technical gap and provides user-friendly communication to extract value insights from graph data. Beyond simplifying query composition, the interpretation of extracted data in an interactive network further eases AOPs network in-depth analysis. This work provides a unified full-stack solution that encompasses essential components i.e., data structure, query generator, and interactive interpretation for AOPs. This tool harmoniously converges to create an invaluable toolset for the AOP developer's community.
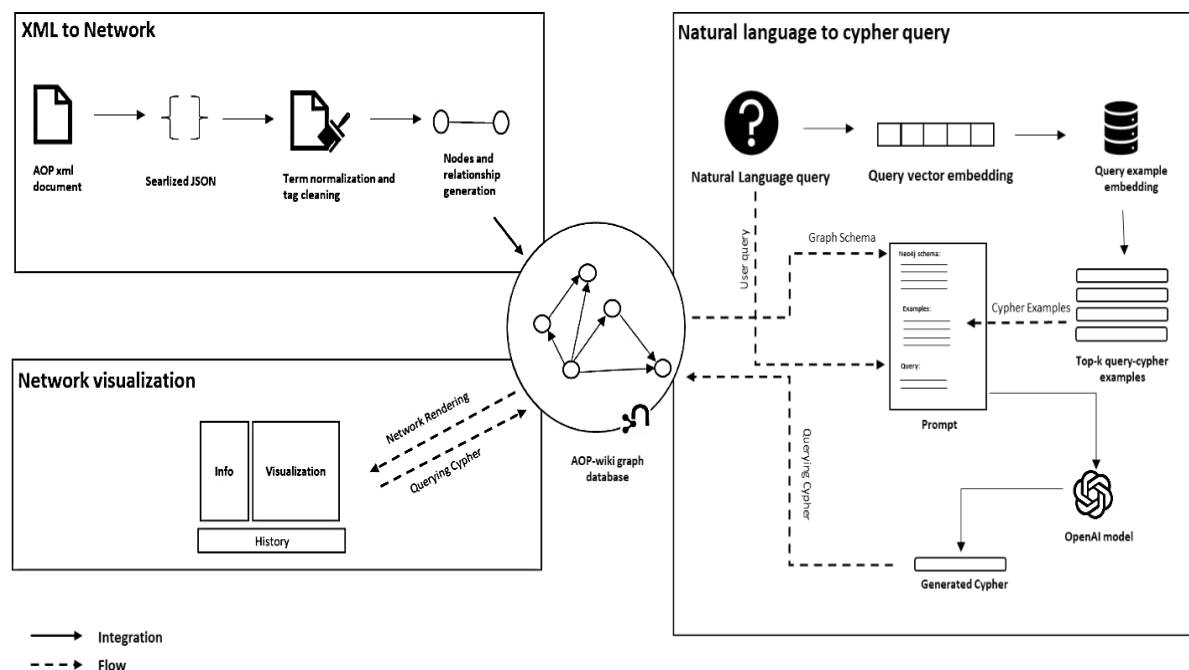
Materials and Methods

Figure 1. Workflow architecture of AOP-Wiki explorer. Each component mentioned in the workflow is developed separately but synchronizes with each other in the operational pipeline. XML to network conversion component processes and compiles the data in into a network form. Schema of compiled AOP-network database is assembled with the prompt to enable natural language to cypher generation process. Finally, the network visualization component retrieves the graph from query and render it in network form. AOP-wiki graph database acts as a central hub facilitating interaction among these components.

The method of adapting AOP wiki data in graph structure, with multiple query interface, is broadly divided into 3 steps i.e., a.) XML to graph conversion, b.) Natural language to graph query and c.) network visualization. All three components seamlessly interconnect to build a cohesive tool. In this interconnection the schema of the generated graph database in step 1. is useful for prompt creation, followed by query generation using the prompt, and finally visual rendering of the network using query. In the subsequent methods section, we will delve deep into these 3 steps.

XML to native graph database conversion
XML version of the latest AOP data released on April,2023 had been downloaded from https://aopwiki.org/downloads. AOP data gets updated quarterly in a year. XML file is then parsed into a Python dictionary object using the xmltodict library[13]. The converted dictionary object consists of elements, such as AOP, key-event, key-event-relationship, chemical, stressor, and taxonomy which are the building blocks of the whole AOP network (Fig 2). These data elements hold unique IDs, which

helps in referencing and restructuring the block to develop the whole AOP network from scratch. For implementing LPG schema Neo4j platform was used [14].
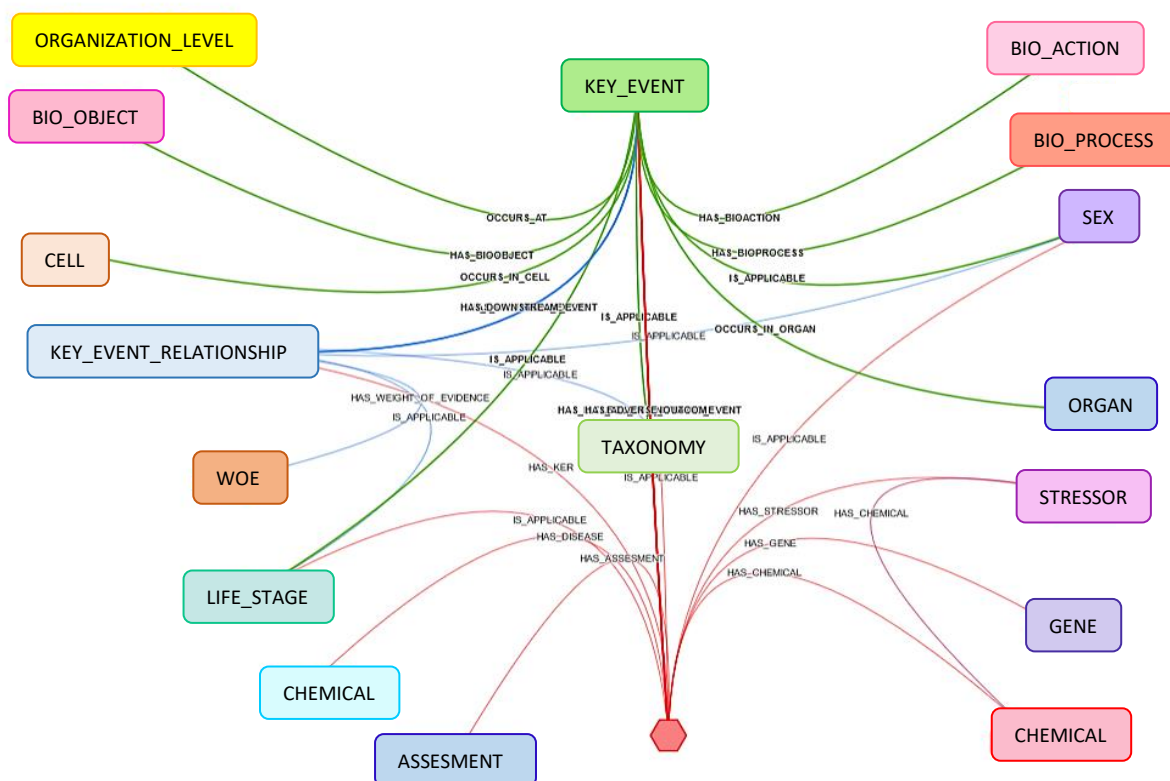

Figure 1. Neo4j graph data model schema for AOP-wiki XML data

Neo4j is a widely used graph database management system, designed to store, manage, and query large amounts of data organized in graph structure. Neo4j relies on cypher query language to interact with graph data. It is designed to express simple to complex queries, that can transverse and retrieve the relationship within the graph. Neo4j, at its core uses Java to run the engine, to make it accessible programmatically Py2neo library was used [15]. Py2neo is a Python library, which acts as an interface between Python and Neo4j and makes programmatic access to neo4j with ease. With Py2neo, nodes, and relationships are built while keeping a consistent naming convention of node and edge labels and their properties. Nodes and edge labels are all in uppercase and the labels having more than one word are joined with the snake naming convention. Properties of nodes and edges are written in lowercase. The strict naming convention provides robustness and consistency in further integration. Nodes, relation, and their properties are tabulated in S-Table 1.

The graph schema in Figure 2. Illustrates the way information within AOP-wiki is structured and represented as a graph data model. Typically, in AOP graphical representation, there are mainly three types of nodes i.e., AOP, key event, and Key event relationship. In this schema, it has been extended to show information like taxonomy, sex, biological organization level, assessment methodology, and many more. Also, while populating the properties of nodes, textual information was processed using regex to clean the HTML tags to make the text human readable. The extended schema gives flexibility to the user to retrieve information at different levels of granularity within AOP, facilitating in-depth analysis. Some of the crucial textual information of AOP such as their assessment methodology and weight of evidence are also represented as nodes to increase information accessibility. The terms like "life stages" and "sex" underwent processing to eliminate redundant information and were standardized by assigning them persistent identifiers from the referenced ontology. From the AOP

wiki portal, sections containing descriptive textual information about each AOP, such as abstract, description, the weight of evidence, and assessment method methodology, etc. have been combined into a single document. Over the combined document Named Entity Recognition (NER) is applied to extract the biological concepts such as gene, protein, chemical, and disease using BERN2 biomedical named entity recognition and normalization tool [16]. Normalized entities are mapped to respective databases and integrated into the network as nodes.

Natural Language to graph query

To bridge the technical gap and provide flexibility to query information in natural language form, an infamous OpenAI's GPT-4 [17]. model was used. OpenAI's GPT-4 is a Large Language model designed to understand and generate human-like text. The GPT-4 model is trained on an extensive and varied corpus of text derived from the internet, encompassing sources like books, articles, websites, documentation, and code repositories. The training on such a massive amount of data enables the model to learn patterns, syntax, semantics, and contextual nuances. GPT-4 model inherently does not possess specific knowledge about Cypher queries, instead, they acquire knowledge through extensive training data or prompts they are exposed to. Description and examples of cypher query provided in the prompt enable the model to generate coherent patterns. However, the response from the model is based on statistical patterns, they have learned while training, rather than a deep understanding of the Neo4j cypher.

The quality and correctness of generated cypher queries depend on how precise the prompt is provided. The precise prompt here signifies explicit instruction, relevant context, illustrative examples of desired responses, and clear specification of the output format. To build precise prompts, in this work dynamic few shot prompt generation methodology has been adopted. In dynamic methodology, few examples in the prompt are provided similar to the query asked by user. Similar examples make the model aware of the syntax and variables to be used for generating the cypher. Similar examples were stored in the form of vector embedding generated by "all-MiniLM-L6-v2" sentence transformer model [18]. Chroma vector database[19] had been used to store examples embedding. This database gets updated with the more examples as query saved by the user. The variable part in the dynamic prompt is the provided example query and query itself, whereas other components are static. Static components of prompts include graph schema, context, and output format. Graph schema holds information about how the database is structured and the properties mentioned in nodes and edges. Neo4j module of langchain[20] was used to generate the schema of graph database. Context provides precise instruction to the model, while output format provides schema such as JSON, or YAML which enable to generate machine parse able cypher.

The overall compilation of prompts follows the sequence mentioned in Figure 1. First, the real-time query of the user is embedded. Based on this query embedding, similar example queries are retrieved using cosine similarity from the vector embedding database. The extracted examples are then merged with other components of prompt and fed to the model for the final cypher query generation.
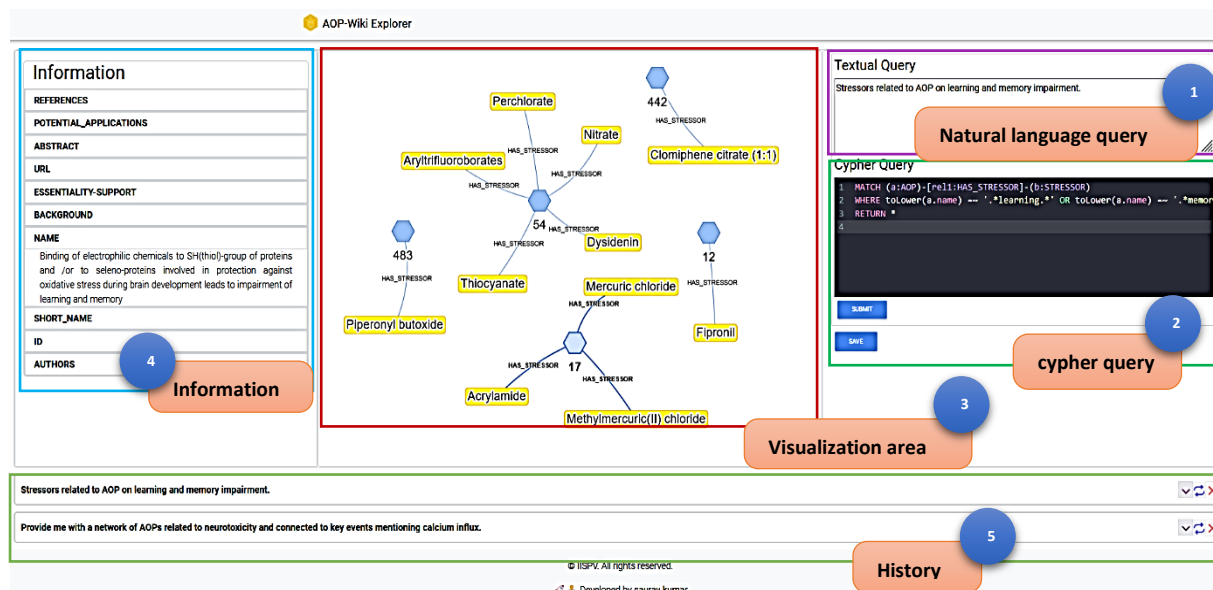
Interactive network visualization

Figure 2. AOP-Wiki Explorer user interface. The AOP Wiki Explorer user interface provides an interactive playground to get graphical insights into AOP. The user interface rationally designs into 4 components i.e., 1.) Natural Language query area, 2.) Cypher query area 3.) Visualization area 4.) Information area 5.) History area

For network visualization, an interactive user interface was built called AOP-wiki explorer. The interface is thoughtfully structured into five distinct components, and each component plays a pivotal role in enhancing user interaction. These components orchestrate harmoniously and provide users with an adaptable playground to analyze AOPs. The components and their role are as follows:

Natural language query: It takes user-initiated queries in natural language form. Input to this component is mandatory as it helps to keep track of human-understandable queries. Additionally, it assists in generating cypher for user who may not possess technical knowledge about cypher.

Cypher query: Takes direct Neo4j cypher query as input. Here users with technical knowledge can craft intricate cypher queries, and able to leverage the full potential of a graph database to extract precise insights and patterns. Syntax highlighting is supported for the cypher which provides readability to the code.

Visualization Area: It is a visual output component that renders the pattern given by the cypher query. Visual representation empowers the users to unravel complex relationships, dependencies, and trends within AOPs. Visualization is powered by the Neovis library [21], which is a part of the Vis JavaScript library specialized in handling Neo4j data.

Information Area: It provides comprehensive details stored as properties of nodes and edges. When tapped over the network component (nodes and edges) in the interactive visualization, it shows the tabular list of properties with detailed information. It also provides the URL to navigate the source of information i.e., AOP-wiki.

History Area: This component allows users to retrace their steps and revisit previous query. This feature facilitates iterative exploration, dynamic data investigation and query merging.

The processing of raw data to the network has been done with Python3 [22]. Neo4J has been used as the graph database platform. Web app has been developed with React, a JavaScript framework for UI development. Along with this other JavaScript library such as Neovis.js, codeMirror.js, etc. has been

used to support other components for visualization. The backend was developed with a Flask micro server in Python. Each component has been constructed as a Docker container, and their coordination is managed through a Docker Compose version 3 file.

Result and Discussion

Adopting AOP-wiki XML as a Graph database

This work adapts AOP-Wiki XML information into an LPG data structure. Transforming XML data to graphs enables the researchers to query AOPs in their more natural way. Along with this AOP wiki information has been further enriched with information about genes/proteins, chemicals, and diseases mentioned in each AOP using Named Entity Recognition. A total of unique 1138 biological entities have been identified, and out of that 439 were gene/protein, 378 chemicals, and 321 diseases. Each of the entities was attached to their respective ontology such as CHEBI is associated with chemicals, MIM is related to disease, NCBIGene for genes, and MESH is a general purpose which covers multiple domains. With ontology association, the problem of duplication of the same entity with a synonym has been resolved. The enrichment allows users to search AOPs with the very fundamental information of their work such as genes or chemicals associated with them and gives more flexibility to information retrieval. The enriched information is available as supplementary information. To make the AOP graph data free of redundant information and FAIRER, information such as life stages and sex has been filtered and attached to its consistent identifier. Out of the 23 identified life stages, redundant ones like "Adults" and "Adult," as well as "Foetal" and "Fetal," have been filtered and attached to their permanent identifier. Similarly, lots of other terms like "Adult, reproductively mature", "1 to < 3 months" and many more were carefully assigned with specific ontology and definition. Terms with unspecified ontology were not changed. FAIRified life stages are available in supplementary.

The schema of the graph database was designed to align with criteria aimed at enhancing information accessibility and interoperability, thus leading to broader applicability of the tool. Information such as life stages, sex, taxonomy, gene/protein chemical, etc. of AOPs and KE often reside deep in their textual descriptions, so rendering and filtering information on these criteria is challenging. Representing this information as separate nodes leads to fine-level granularity, reduced term redundancy, and wider accessibility of information. Additionally, representing KERs as edge property limits the accessibility and interconnectivity with other vital details such as sex, life stages, etc. To address this concern, multiple schemas were designed and the one which meets the criteria was selected as shown in Figure 2., the network visuals of all other tested schemas are present in the S-Figure 2,3.

Adapting graph database schema is particularly well suited for biomedical domain due to its inherent complexity and numerous interconnected sub-domains within it. Storing biomedical information in relational paradigm (SQL/NON-SQL) can be understood based on its technological maturity in the database field. However, data such as AOPs contain many-to-many, densely connected relations and querying these databases requires resource intensive JOIN operation in relational databases which impact the performance as the data quantity grows. Although the tabular data can be modeled in graphs but optimizing complex queries and graph algorithms like path traversals are challenging without complementary adjacency list. Whereas graph database system such as Neo4j offers direct interaction with native graph model, eliminating manual relationship engineering. They provide straightforward methods to access, curate and operate on data while also offering flexibility to adapt and evolve with the introduction of new data or schema. Many prominent biological databases have undergone a transition to embrace the features of graph databases, which has proven advantageous

for their data management, query capabilities, and the ability to integrate information from various external sources. Notable databases such as Reactome [1,23] have adopted this approach.

Addressing query crafting with LLM integration

Figure 4. Query complexity comparison between cypher and SPARQL. Both queries retrieve the same information about the chemicals which leads to adverse outcomes of fibrosis.

Each database comes up with its own syntax or query language to facilitate efficient information retrieval. SPARQL Protocol and RDF Query Language SQL (SPARQL) are used for managing??? and retrieval from databases mapped in RDF format. Similarly, the cypher query is used with Neo4j graph databases. In terms of complexity, the neo4j query is far less complex and more intuitive than the SPARQL query. As an example, to retrieve the chemicals which are related to "fibrosis" as an adverse outcome, the syntax mentioned in Fig 4. is used, to retrieve the same information with both the cypher and SPARQL query. The cypher query pattern resembles to graph, where the parenthesis "()" represents the node, while "-[]->" represents the edge. Filtering the nodes and relationships based on certain criteria can be done using the WHERE clause or directly imputing property as key and value in the nodes and relationship. Static clauses used in cypher and SPARQL both are comprehensible and represent the intended process.

Natural Language Query	Visual representation of network generated from query
Simple: Stressors related to AOP on learning and memory impairment.	

Moderate: Provide me with a network of AOPs related to neurotoxicity and connected to key events mentioning calcium influx.	
Complex: common key events connecting the AOP related to neurotoxicity	

Table 1. Different variations of query simple, moderate, and complex are shown in natural language and network format. Cypher query for each natural language query is present in the supplementary.

Querying connected information from graph databases provides significant flexibility, the flexibility entails the inherent nature of graph databases. Graph databases consider the relation between data points as first-class information in the data model, which is crucial for biological information. Mostly, the biological facts are non-linear, one fact influences other facts in multiple ways, and the same is applicable to AOPs as well. Graph transversal is one of the key properties of graph databases, it makes it efficient to navigate from one node to another and capture the intermediate relationship between them.

Depending on the research question, the complexities of database queries can range from relatively simple to highly complex. Table 2??? and Figure 4 illustrate the diversity of questions and queries researchers may need to address while building or evaluating the AOP. Crafting complex queries is not an easy task for researchers from non-technical backgrounds. The advent of Large Language models (LLM) emerges as a valuable resource and offers a solution to tackle the problem of crafting queries. With a few examples of natural language and cyphers in the form of a prompt, the model adapts the pattern and syntax of cyphers being able to generate the cyphers for real-time queries asked by the user. LLM acts as a wrapper and bridges the gap of technical needs. Well-curated natural language query in accordance with the schema of a graph database is enough to generate precise cypher. Having no communication barrier between the information source and the users will increase the

accessibility and usability of the tool. The growing usability of the tool results in the addition of diverse queries in the AOP context from the users leading to an increase in precision and accuracy of cypher generation. The diverse queries asked by the developers will also reflect the needs and expectations they have from the AOP wiki; hence this will be also helpful in modulating the AOP wiki as per the user's requirement in future updates.

In this study we used OpenAI GPT-4 with 8k context size closed source LLMs, for each request to the language model charges $0.03/1k tokens. On average our prompt consists of 5k tokens, which costs us $0.15 for each natural language to cypher generation. These expenses are anticipated to reduce over time, as the database accumulates relevant examples, we expect to rely on prompt examples resembling user queries, thereby reducing the need of imputing multiple examples. Furthermore, with the rapid advancement of large language model development in the open-source community, the problem concerning cost will be addressed. The recent release of open open-source LLMA model by Facebook shows the capacity to generate code and has garnered attention. In the future fine-tuning models like LLMA on cypher queries will be a cost-effective alternative to the OpenAI model.

Interactive and stepwise query

The interface for exploring AOP development is designed to facilitate interactive and stepwise queries, acknowledging the inherently exploratory nature of AOP development. In this section, we will explain the things with the help of an example query i.e. "What are the key events that are connecting the AOPs which is applicable to fish and rat taxonomy and has adverse outcome related to fertility or reproductive issue". From the query, you can infer that the complex query can be broken down into multiple queries such as 1.) AOPs which are applicable to "fish" and "rat" 2.) Filter the AOPs, which mention "fertility" or "reproductive" issues in the adverse outcome 3.) Find the shortest path between the filtered AOPs which are connected by key events only. The above-mentioned queries are rationally divided into steps and queried separately, which allows users to analyze the results of the query individually and make further decisions as per requirement. In Figure 5. above mentioned natural language query is presented in cypher query and the visual representation of the query is present in supplementary.

Future perspectives

The work presented herein provides an enhanced ability to computationally query, retrieve, and integrate AOP-Wiki's data by converting it into a more amenable graph format, and by leveraging large language models to expose the internal representation of the AOPWiki database in a formulation more natural to AOPs' formulation. However, the machine actionability of AOP-Wiki remains limited. This is due to the incomplete semantic definition and functional linking of its internal representation. While, as previously discussed, there have been efforts to transform AOP data into RDF, its natural formulation remains that of a relational database, with limited linkage between its internal entities and external semantic definitions. AOPWiki and associated projects provide limited ontology-like structures, however these are not strictly defined, for example with species hierarchies not adhering to subsumption, meaning that e.g., information concerning mice may be split across distinct subsets of the ontology classified as "mouse," "mouse and human," "mouse and zebrafish and human.". Outside the representation of its structured data, AOPWiki data are also highly multi-modal, with much of the valuable and specific information being expressed in text or images, which are inherently unstructured or semi-structured.

Importantly, the ability of large language models to make sense of this representation is limited, and its performance in tasks involving identifying semantic entities from text or databases is limited, below those of other state of the art methods (cite). Issues with hallucination, especially at the edges of knowledge are also well documented. These issues are compounded where data are novel, on the

edge of the scientific frontier, and therefore they cannot in this context be relied upon as an alternative for serious, collaborative efforts to producing high quality, FAIR, functionally linked data. We therefore see a more comprehensive approach to developing a fully semantic representation to be the next step in fully realising the potential of AOPs.

The value of building a fully semantic implementation of AOPWiki inheres in the ability to computationally integrate the knowledge expressed in AOPs with knowledge concerning the relevant entities across the scientific knowledge landscape. Increasingly, data concerning biomedical and chemical entities are made public in linked representations, most often using OWL ontologies, which provide the ability for semantic confirmation of equivalence of entities across databases, to associate AOP events with external scientific evidence, and provide a background formal semantics that can be used to ensure consistency, and to directly support analysis with background knowledge. In addition to simple semantic equivalence, the description logics that underpin biomedical ontologies can be used to support the development of formal modelling approaches that will be required to use AOPs in risk assessment.

The process of realising this vision, however, will take substantial work. It is clear from our work that the RDF representation of AOP data is not entirely natural, and this will require innovation to align a more natural representation of AOPs with representations found elsewhere in the chemical and biomedical domains, which are mostly expressed in triples. Furthermore, while AOPWiki currently provides the ability to annotate its internal entities, such as events, with ontology terms, the nature of those relationships are not well explicated, and are therefore not machine actionable. We envision a future with fully functionally linked evidential associations for AOPs. However, the data and metadata structures for these entities do not yet exist. While solutions such as BioPAX and BioLink would provide the features required to express associations themselves, they will require additional work to support functional integration with AOP data structures, as well as the provenance, authorship, levels of confirmation, and other relevant metadata. These would also support efforts to perform information extraction, linkage, and integration from text and other unstructured or semi-structured data sources, both from within the AOPWiki database, as well as from other sources such as literature or policy documents. Inspiration may be drawn from other biomedical projects, such as the Gene Ontology project, whose instance databases include evidence codes. A general solution to evidence would support a wide range of uses across science, for example in the area of disease-phenotype associations and their use in biomedical research and healthcare.

Query 1.

AOPs applicable to "fish" and "rat" taxonomy?

Query 2.

Filter the AOPs, which mention "fertility" or "reproductive" issues in the adverse outcome

Query 3.

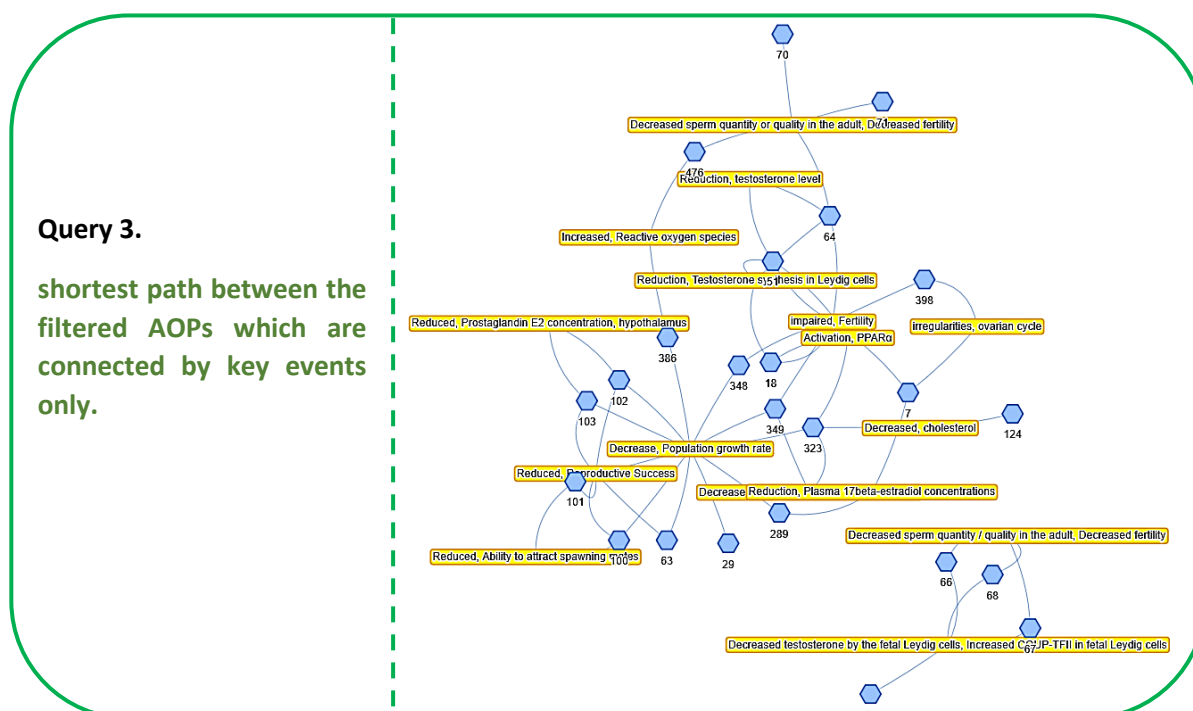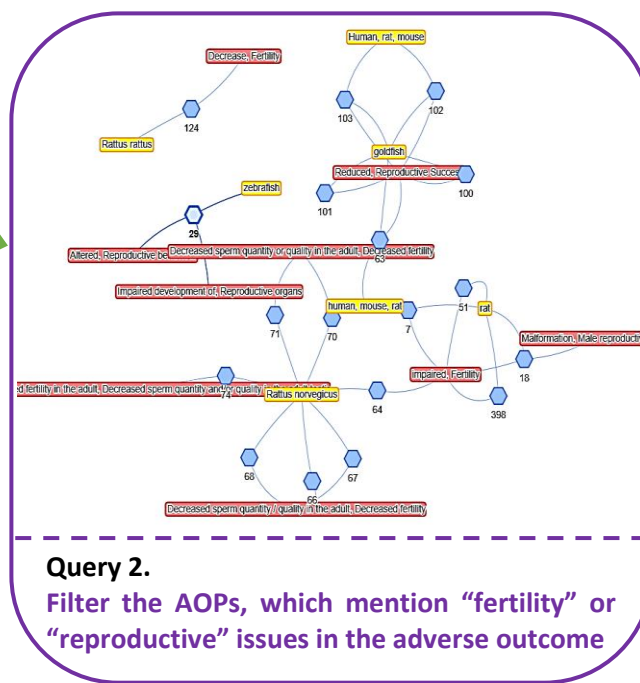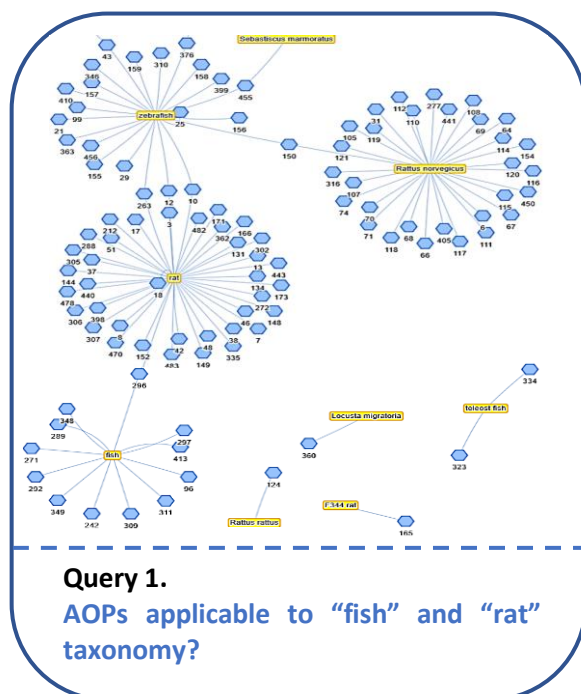shortest path between the filtered AOPs which are connected by key events only.

Figure 3. Network view of step wise query

Conclusions

In this work we represent AOP-wiki data as a graph and build tools to enable the users to query and link information in a very flexible way. In this work, we fill the technical gap and implement the large language model's ability to generate cypher queries based on the natural language query provided in the context of a graph database schema. Having an exploratory nature to the AOP development process, we thoughtfully designed the interface which allows users to retrieve their complex research queries in multiple steps and keep track of individual queries separately. The results of the queries will be retrieved in interactive network form. The Detailed information on AOPs, key events, and their relation is accessible in tabular form and provides the functionality to retrace the information source.1 paragraph can be written here.

Acknowledgments

Conflicts of Interest Statement:

There is no conflict of interest. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the Health and Digital Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

Code and software:

The model code for AOP wiki explorer is available on GITHUB with the following link.

References

1   G. T. Ankley, R. S. Bennett, R. J. Erickson, D. J. Hoff, M. W. Hornung, R. D. Johnson, D. R. Mount, J. W. Nichols, C. L. Russom, P. K. Schmieder, J. A. Serrrano, J. E. Tietge, D. L. Villeneuve, *Environ Toxicol Chem* **2010**, *29*, 730.

2   D. Knapen, M. M. Angrish, M. C. Fortin, I. Katsiadaki, M. Leonard, L. Margiotta-Casaluci, S. Munn, J. M. O'Brien, N. Pollesch, L. C. Smith, X. Zhang, D. L. Villeneuve, *Environ Toxicol Chem* **2018**, *37*, 1723.

3   C. Wittwehr, L.-A. Clerbaux, S. Edwards, M. Angrish, H. Mortensen, A. Carusi, M. Gromelski, E. Lekka, V. Virvilis, M. Martens, L. Olavo, B. Da, S. Santos, P. Nymark, *ALTEX - Alternatives to animal experimentation* **2023**.

4   Users' Handbook supplement to the Guidance Document for developing and assessing Adverse Outcome Pathways | OECD Series on Adverse Outcome Pathways | OECD iLibrary, https://www.oecd-ilibrary.org/environment/users-handbook-supplement-to-the-guidance-document-for-developing-and-assessing-adverse-outcome-pathways_5jlv1m9d1g32-en.

5   C. Vicknair, X. Nan, M. Macias, Y. Chen, Z. Zhao, D. Wilkins, *Proceedings of the Annual Southeast Conference* **2010**.

6   C. Ives, I. Campia, R.-L. Wang, C. Wittwehr, S. Edwards, *Appl In Vitro Toxicol* **2017**, *3*, 298.

7   M. Martens, C. T. Evelo, E. L. Willighagen, *Appl In Vitro Toxicol* **2022**, *8*, 2.

8   H. M. Mortensen, M. Martens, J. Senn, T. Levey, C. T. Evelo, E. L. Willighagen, T. Exner, *Frontiers in Toxicology* **2022**, *4*, 803983.

9   M. D. Wilkinson, M. Dumontier, Ij. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J. W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. G. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. C. t Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S. A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. Van Der Lei, E. Van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, B. Mons, *Scientific Data 2016 3:1* **2016**, *3*, 1.

10   S. Timón-Reina, M. Rincón, R. Martínez-Tomás, *Database* **2021**, *2021*, 26.

11   J. Barrasa, RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?, https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/.

12   E. Cucen, Making Sense of Data with RDF* vs. LPG - OpenCredo, https://opencredo.com/blogs/making-sense-of-data-with-rdf-vs-lpg/.

13   PyPI, XML2Dict, https://pypi.org/project/XML2Dict/.

14   Neo4j, https://neo4j.com/.

15   PyPI, Py2neo, https://pypi.org/project/py2neo/.

16   M. Sung, M. Jeong, Y. Choi, D. Kim, J. Lee, J. Kang, *Bioinformatics* **2022**, *38*, 4837.

17   OpenAI, **2023**.

18   N. Reimers, I. Gurevych, *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference* **2019**, 3982.

19   Chroma, https://docs.trychroma.com/.

20   LangChain, https://python.langchain.com/docs/get_started/introduction.

21   Neovis.js, https://github.com/neo4j-contrib/neovis.js?

22   G. Van Rossum, F. L. Drake, *Scotts Valley, CA* **2009**, 242.

23   M. Gillespie, B. Jassal, R. Stephan, M. Milacic, K. Rothfels, A. Senff-Ribeiro, J. Griss, C. Sevilla, L. Matthews, C. Gong, C. Deng, T. Varusai, E. Ragueneau, Y. Haider, B. May, V. Shamovsky, J. Weiser, T. Brunson, N. Sanati, L. Beckman, X. Shao, A. Fabregat, K. Sidiropoulos, J. Murillo, G. Viteri, J. Cook, S. Shorser, G. Bader, E. Demir, C. Sander, R. Haw, G. Wu, L. Stein, H. Hermjakob, P. D'Eustachio, *Nucleic Acids Res* **2022**, *50*, D687.