An Interactive Knowledge and Learning Environment in Smart Foodsheds

Yamei Tu, Xiaoqi Wang, Rui Qiu, Han-Wei Shen
The Ohio State University

Michelle Miller, Jinmeng Rao, Song Gao
University of Wisconsin-Madison

Patrick R Huber, Allan D Hollander
University of California Davis

Matthew Lange
International Center for Food Ontology Operability Data and Semantics (IC-FOODS)

Christian R Garcia, Joe Stubbs
The University of Texas at Austin Texas Advanced Computing Center

***Abstract*—The Internet of Food (IoF) is an emerging field in smart foodsheds, involving the creation of a knowledge graph (KG) about the environment, agriculture, food, diet, and health. However, the heterogeneity and size of the KG present challenges for downstream tasks, such as information retrieval and interactive exploration. To address those challenges, we propose an interactive knowledge and learning environment (IKLE) that integrates three programming and modeling languages to support multiple downstream tasks in the analysis pipeline. To make IKLE easier to use, we have developed algorithms to automate the generation of each language. In addition, we collaborated with domain experts to design and develop a dataflow visualization system, which embeds the automatic language generations into components and allows users to build their analysis pipeline by dragging and connecting components of interest. We have demonstrated the effectiveness of IKLE through three real-world case studies in smart foodsheds.**

THE INTRODUCTION

Knowledge graphs provide a powerful means to capture and integrate diverse linked knowledge into a unified graph structure [1]. Many knowledge graphs have been developed to support a range of industrial and scientific applications, including the Google Knowledge Graph, DBpedia [2], Wikidata [3]. In the context of smart foodsheds, there is a growing interest among researchers from industry, academia, and government in building a semantic platform for the Internet of Food (IoF). The IoF represents a linked domain knowledge of the environment, agriculture, food, diet, and health [4], [17]. By using the knowledge graph to link food system stakeholders with data, cyberinfrastructure, AI, and other tools, the IoF can improve the current food system for increased food sustainability, health, justice, and delight. It also facilitates many downstream tasks, such as information retrieval and food flow data analysis in food systems [17], [20].

The knowledge graphs in smart foodsheds face several challenges for downstream tasks due to their massiveness and heterogeneity. First, specialized food vocabularies and ontologies used by different organizations require a unified environment for data exchange and information sharing, but defining standardized ontologies for different downstream tasks is not trivial. Second, knowledge graphs are commonly built from ontologies and represented as Resource Description Framework (RDF) triplets. Structured query languages like SPARQL can query RDF triplets, but writing SPARQL queries can be challenging for users unfamiliar with the underlying ontologies. Hence, easy-to-use query methods are urgently needed. Lastly, after retrieving relevant information from knowledge graphs, identifying data patterns is crucial to facilitate the decision-making process. Automatic visualization generation can efficiently reveal data patterns and draw meaningful insights.

Many technologies and systems have been proposed to address the individual challenge mentioned above. Protégé is the most popular ontology editor in the world for ontology development. But it only focuses on the ontology definition and does not address how to utilize it. To query the knowledge graph, pre-trained language models like BERT [5], GPT [6], and RoBERTa [7] can be fine-tuned with sequence-to-sequence tasks to translate natural languages to the SPARQL. While Neo4j allows users to interact with data through visual interactions, it only supports node-link diagrams for visualization. Thus, there is a research gap in developing a unified environment that addresses various challenges, including incorporating ontologies for data exchange, automated query generation, and visual analytics of knowledge graphs.

This work introduces an Interactive Knowledge and Learning Environment (IKLE) to address the research gap. The IKLE leverages three different languages, including Linked Data Modeling Language (LinkML), SPARQL, and Vega-Lite, to accomplish ontology building, query generation, and visualization recommendations while keeping users in the loop. Collaborating with domain experts, a dataflow system is designed and developed to demonstrate the flexibility of IKLE for interactive exploration. The dataflow system modularizes each functionality into components, enabling users to build their analysis pipeline according to their skill level and verify the output of each step. We evaluate the proposed method through case studies in the smart foodshed literature. In short, the contributions of this work can be summarized as follows:

- Modeling the knowledge graph interactions as a combination of three programming and modeling languages and generating them on demand based on user interactions.
- Proposing a general solution, IKLE of KG interactions, which can be easily extended and transferred to other domains.
- Designing a dataflow system that enables users to build their analysis pipeline according to their needs.
- Demonstrating the usefulness and effectiveness of our system qualitatively through case studies in smart foodsheds with input from domain experts for further improvements.

Related Work

Dataflow Visualization System

Dataflow systems allow users to create customized functionality by building a dataflow diagram that outlines how components interact with each other. These systems can be divided into two main categories: general computational [21] and visualization systems. For the purposes of this work, we focus on dataflow systems that specialize in data manipulation and visualization. VisFlow [11] is a web-based visualization framework for tabular data which employs a subset data flow model, allowing interactive queries within the data flow. Based on it, VisFlowj [12] is an enhanced subset-flow visualization system that integrates natural language processing techniques to facilitate multi-view visualization. ExPlates [13] is another system developed to allow users to manipulate data and build visual queries to explore multidimensional data. Viscomposer [14] is a programmable integrated development environment (IDE) to make visualization design easier with its intuitive user interface. To the best of our knowledge, While many existing dataflow systems analyze tabular data, few of them address the challenges of KG exploration.

This study proposes an interactive knowledge and learning environment for KG exploration in smart foodsheds, which can be easily extended to other domains.

Knowledge Graph Interaction

The knowledge graph (KG) has attracted much attention in both academia and industry recently. Various systems have been built to facilitate sub-tasks of KG in different domains. For the knowledge reasoning task, many clinical decision support systems incorporate knowledge graphs to obtain valuable medical knowledge and diagnose evidence [15]. As for link prediction, by modeling user-product interactions with external knowledge as a knowledge graph, the recommendation systems are trained to predict items of interest for new coming users [16]. Some commercial tools are released to support querying and visualizing the knowledge graph, such as Neo4j [1], Gruff [2]. In this work, we propose a general environment to make progress on enabling a variety of tasks necessary for smart foodsheds, such as cohering domain ontologies into application data models through LinkML, SPARQL query generation, and visualization of either raw data or output of models.

Internet of Food

Originally, the concept of the Internet of Food (IoF) first appeared in 2011, which is mainly concerned with issues related to how technology impacts the food system. Later on, since food systems are composed of a diverse range of social and natural components, IoF represents a linked heterogeneous knowledge (knowledge graph) of data describing these components coming from numerous sources[4]. Recently, there are several works aiming at analyzing knowledge graphs in food systems but with different focuses. For example, [20] analyzes the resilience of food networks at different geographic scales via the technique of geospatial knowledge graph. [18] utilizes a large-scale food knowledge graph to recommend food according to dietary preferences and health guidelines. In this work, we develop a visualization system to help users interactively explore the IoF and thus facilitate some potential downstream tasks, including partner finder, information retrieval, resilience analysis and etc.

Preliminaries

First, we introduce the PPOD knowledge graph that we used in this paper, and then we formalize the problem we aim to address.

Dataset

To address information inefficiencies in the food system, experts first developed the ontology of PPOD, a schema that describes the attributes and relationships between **P**eople, **P**rogram, **O**rganizations and **D**ata sets. To instantiate this ontology with real data, they built a PPOD knowledge graph using data from California and Ohio as a first use case.

The PPOD knowledge graph $\mathcal{G}$ is represented as a set of triplets $\mathcal{G} = \{(s, p, o)\}$ that describes the relationships between different entities in the food system. Each triplet (s, p, o) characterized the semantic relationships p between source entity s and object entity o. For example, the triplet ("Cosumnes River Project," "leading_organization," "The Nature Conservancy") means that *The Nature Conservancy* is the leading organization of the *Cosumnes River Project*. Entities in the triplets are represented using either URIs (universal resource identifier) $u \in \mathcal{U}$ or literal $l \in \mathcal{L}$. Each subject s must have a unique URI $s \in \mathcal{U}$, while each object o can be either URI or literal $o \in (\mathcal{U} \cup \mathcal{L})$. Entities represented as literal can only be the tail of a triplet and are denoted as $\mathtt{E}_L$. Entities with URIs can either be the head or tail of a triplet and are denoted as $\mathtt{E}_U$. These definitions are used consistently throughout the paper.

Problem Statement

The goal of this work is to improve information access in smart foodsheds by leveraging visual analytics to facilitate access to the PPOD knowledge graph. To achieve this, our key tasks involve defining the ontology $\mathcal{G}_{ont}$ in a machine-understood way, using $\mathcal{G}_{ont}$ and user query q to retrieve information from the corresponding PPOD knowledge graph $\mathcal{G}$, and visualizing the resulting valid triplets $\mathcal{G}' = (s, p, o|q)$ from $\mathcal{G}$ in an intuitive manner to aid decision-making.

[1] https://neo4j.com/
[2] https://allegrograph.com/products/gruff/

This article has been accepted for publication in IEEE Computer Graphics and Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCG.2023.3263960

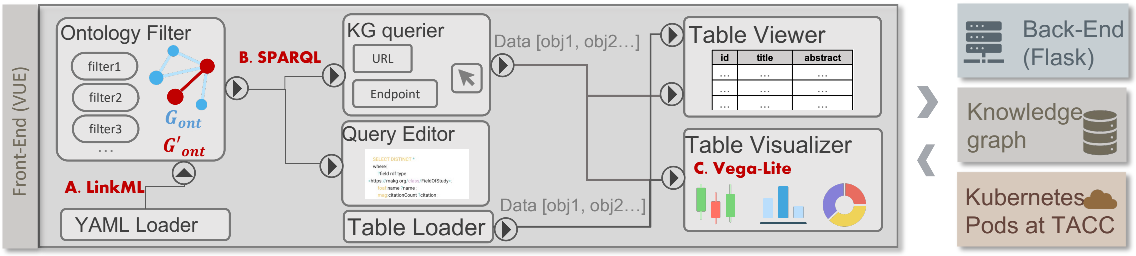


Figure 1: Overview of our system and its architecture.

Requirement Analysis

To gain a better understanding of the needs of domain users in smart foodsheds, we held weekly meetings with the domain experts of this project for over one year. During these meetings, we derived the system's requirements from the discussions as follows:

R1. Incorporating expert-defined ontologies. Our system should enable users to easily load and utilize different expert-defined ontologies for downstream tasks, facilitating knowledge sharing as suggested by our experts.

R2. Easy querying of knowledge graphs. As pointed out by the experts, structuring SPARQL queries from scratch can be difficult for non-programming users. Thus, our system should provide guidance to help users prepare queries for retrieving information from knowledge graphs. Specifically, it should:

- *R2.1* Automatically generate SPARQL queries to ease the burden on users who are unfamiliar with the structured language's schema and rules.
- *R2.2* Allow users to fine-tune the automatically generated queries to increase transparency and trust in the system.

R3. Efficient data visualization. Designing an effective mapping from data to visual channels requires specialized skills and resources that may not be readily available to all users. Thus, our system should provide recommendations for data visualization to help users easily understand and interpret the data.

R4. Personalized analysis pipeline building. Our domain experts mentioned that users with different backgrounds and skill levels might have varying needs, and a system with a pre-defined pipeline cannot meet all of these needs. Thus, our system should enable users to build the analysis pipelines flexibly to serve a broader audience and meet a wider range of needs.

Method

The requirements have driven us to develop an interactive knowledge and learning environment (IKLE) to support various interactions with the knowledge graphs. In this section, we first present an overview of IKLE and then introduce each of its components in more detail.

Overview

The key challenge behind IKLE is how to leverage the machine's ability to handle various tasks and reduce the burden of domain users. We propose to identify three key programming languages (PLs) that are involved in knowledge graph interactions. One advantage of using these PLs is that machines can process and understand them, which enables us to automate each subtask. In summary, our method consists of four components, summarized as follows:

- LinkML of ontology definition (Figure 1A): We first describe the key design of the ontology, which can be further parsed into an ontology graph to help users to understand the knowledge graph structure (**R1**).
- SPARQL generation (Figure 1B): Users can interact with the ontology graph and build the query interactively, triggering the automatic generation of SPARQL (**R2**).
- Vega-Lite generation (Figure 1C): Once users query data from knowledge graphs, we automatically recommend visualization and create Vega-lite specification for chart rendering (**R3**).

This article has been accepted for publication in IEEE Computer Graphics and Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCG.2023.3263960

- dataflow system: A dataflow system that is developed as an implementation of the IKLE. It achieves high flexibility by allowing users to build their own analysis pipeline(**R4**).

Ontology Design and Parse

The ontology defines the different types of entities and their relationships with each other. This structural information is critical for users to understand the information provided by the knowledge graph and for machines to generate SPARQL queries automatically (**R1**). However, it can be challenging to design an ontology schema that (1) includes all necessary information for creating queries; (2) can be easily parsed and visualized for interactive queries; (3) has a clear structure that can be adapted to other domains.

We have collaborated with domain experts to identify the key information required for the ontology, resulting in the following data definition:

- *Entity Types* ($\mathbb{T}_E$): This defines the types of entities with URI $\mathtt{E}_U$, with each entity type $\mathtt{T}_e \in \mathbb{T}_E$ having a defined URI ($\mathtt{T}_e$.URI) and a set of relations ($\mathtt{T}_e$.rel) starting from $\mathtt{T}_e$.
- *Relation Types* ($\mathbb{T}_R$): To reduce information redundancy, each relation type $\mathtt{T}_r$ is defined first and referred to by a unique identifier when defining it in $\mathtt{T}_e$.rel. $\mathtt{T}_r$.name provides the semantic meaning of the relation, while $\mathtt{T}_r$.URI indicates the relation type's URI. $\mathtt{T}_r$.targets defines the targeting entity types of this relation type.
- *Filters* ($\mathbb{F}$): A set of filter conditions on different entity types $\mathtt{T}_e$ are pre-defined by domain experts to help users identify relevant information. Each filter $\mathtt{f} \in \mathbb{F}$ contains an entity type $\mathtt{T}_e$ and a set of permissible entities with their URIs, denoted as$\mathtt{f} = (\mathtt{T}_e, \{u_e^1, u_e^2, .., \})$.

LinkML is a flexible modeling language that specifies data schema in YAML and can be translated to other schema representation formats, such as JSON or RDF. Due to such flexibility, we utilize LinkML to describe the PPOD ontology, which can be parsed by machines into a graph structure $\mathcal{G}_{ont}$, allowing users to interact easily.

The ontology graph $\mathcal{G}_{ont}$ is constructed using a parsing algorithm. Each node in $\mathcal{G}_{ont}$ represents an entity type $\mathtt{T}_e$, while the edge indicates the semantic relationship $\mathtt{T}_r$ between different entity types. The input to the algorithm is a LinkML file that contains three aspects of information: entity types $\mathbb{T}_E$, relation types $\mathbb{T}_R$, and filters $\mathbb{F}$. For each entity type $\mathtt{T}_e \in \mathbb{T}_E$, we iterate over its outgoing relations $\mathtt{T}_e$.rel and check whether it is connected to another entity type $\mathtt{T}'_e$. If so, we add the edge $\mathtt{T}_e \rightarrow \mathtt{T}'_e$ and nodes $\mathtt{T}_e, \mathtt{T}'_e$ to $\mathcal{G}_{ont}$. Since only entities with URIs $\mathtt{E}_U$ are defined in $\mathbb{T}_E$, we create a conceptual node *literal* in $\mathcal{G}_{ont}$ to represent all literal entities $\mathtt{E}_L$, e.g., the "email" of a "Person." Additionally, we add an attribute, called *filtered* when adding nodes $\mathtt{T}_e$ into $\mathcal{G}_{ont}$. The attribute indicates whether $\mathtt{T}_e$ has definitions in the filter $\mathbb{F}$ and helps differentiate nodes with/without filtering conditions when visualizing the ontology graph for users.

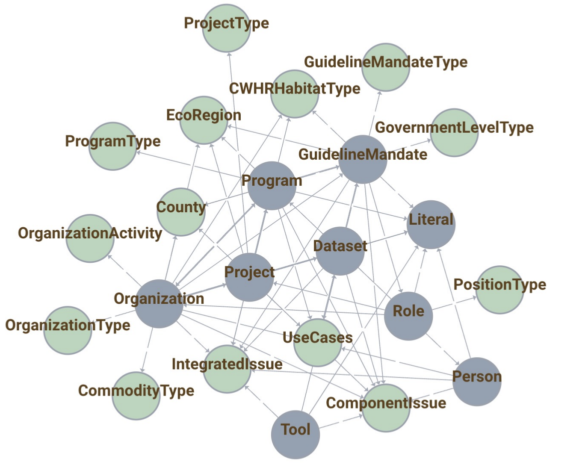


Figure 2: Ontology graph $\mathcal{G}_{ont}$ of PPOD knowledge graph.

The resulting $\mathcal{G}_{ont}$ for PPOD is shown in Figure 2, with literal entities $\mathtt{E}_L$ aggregated into one conceptual node and $\mathtt{E}_U$ that are not defined in $\mathbb{F}$ shown in gray (●), while $\mathtt{E}_U$ that have predefined vocabularies in $\mathbb{F}$ are colored in green (●). This allows users to select a set of desired filtering conditions.

Querying Knowledge Graph

While building visual queries has been extensively studied in the visualization literature, they cannot be applied directly to smart foodsheds data because our target users are not experienced in visual programming and are more used to traditional interactive methods. Moreover, according to our domain experts, building queries from

This article has been accepted for publication in IEEE Computer Graphics and Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCG.2023.3263960

scratch is less effective than filtering a subset of existing ontology. To better assist users in building queries (**R2**), we employ a top-down method that involves presenting the $\mathcal{G}_{ont}$ to users and allowing them to select a sub-graph $\mathcal{G}'_{ont}$ through various interactions such as clicking, brushing, and dragging. SPARQL queries are generated based on the selected sub-graph $\mathcal{G}'_{ont}$.

We have summarized common graph patterns of $\mathcal{G}'_{ont}$ based on real-world use cases and provided example queries in Table 1. Green nodes (●) represent entities with filtering conditions, while gray nodes (●) do not. Dashed lines indicate the existence of one/multiple filtering nodes. For example, in the first row of the table, users query *organizations* located in a specific *ecoregion*.

When generating the SPARQL queries based on the $\mathcal{G}'_{ont}$, the main challenge lies in constructing the `WHERE` clause due to the complexity of constraints. The `WHERE` clauses should strictly follow the ontology while also specifying the filtering conditions on entity types. To address this challenge, we first define four types of subqueries in the `WHERE` clause and aggregate them according to the entity types in $\mathcal{G}'_{ont}$. **Declaration()** defines the entities $\mathtt{E}_U$ through its `URI` to locate the information of interest. Additionally, **Attribute()** detects which literal entities $\mathtt{E}_L$ are connected to a specified $\mathtt{E}_U$ and adds them to the `WHERE` clause. **Relation()** defines the triplet relation, i.e., $\mathtt{T}_e$ is connected to $\mathtt{T}'_e$ through semantic relation $\mathtt{T}_r$. **Filter**() constrains $\mathtt{T}_e$ to a user-selected conditions u. As indicated in the Algorithm 1, we iterate over each edge (s, p, o) in the $\mathcal{G}'_{ont}$ $(line_1)$ and generate the query accordingly, depending on whether the nodes s/o has filtering conditions through an attribute `filtered` $(line_{3,9,14})$. If there is one filtering node, the *Filter()* is required $(line_{12,17})$. Otherwise, we only need to define the subject s, object o and its predicate p $(line_{3-7})$.

Algorithm 1 SPARQL Generation

Input: User-selected ontology subgraph $\mathcal{G}'_{ont}$
Output: SPARQL query Q

```
 1: for (s, p, o) ∈ $\mathcal{G}'_{ont}$ do
 2:     if !s.filtered & !o.filtered then
 3:         $Q$+=Declaration(s), $Q$+=Attribute(s)
 4:         $Q$+=Declaration(o), $Q$+=Attribute(o)
 5:         $Q$+=Relation(r)
 6:     else if s.filtered & !o.filtered then
 7:         $Q$+=Declaration(o), $Q$+=Attribute(o)
 8:         $Q$+=Relation(r)
 9:         $Q$+=Filter(s)
10:     else if !s.filtered & o.filtered then
11:         $Q$+=Declaration(s), $Q$+=Attribute(s)
12:         $Q$+=Relation(r)
13:         $Q$+=Filter(o)
14:     end if
15: end for
```

Visualization Recommendation

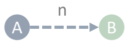
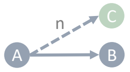
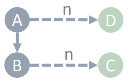
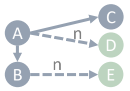
To gain informative insights from the knowledge graph, scanning through the queried results row by row is much less effective than visualizing the queried results with proper visualization charts. We exploit rule-based visualization recommendations to achieve better explainability (**R3**). Taking the tabular data items as input, we transform them to the Vega-Lite grammar, including both visual elements specifications and data items. The Vega-Lite specification can later be rendered as charts and allow users to download them as figures. We build the visualization recommendation module as a python library[3] based on the implementation of VizKG [19], which converts tabular data to charts using Plotly.

The recommendation contains several steps. First, to display the input data patterns properly, we identify the data types of each column via regex matching. Currently, the supported datatypes include numerical, date, and categorical. Then, based on the mapping rules[4], we check the supported charts with their specified datatypes and identify valid charts as *candidates*. The validated charts follow several rules: (1) the data type of the input data must conform to the required data types of the chart. (2) if there is more than one conforming data type, we take a first-come-first-map strategy. The supported chart types include histograms, scatter plots, line charts, box plots, area charts, maps, and donut charts.

Data Flow Visualization System

To create the IKLE, we develop a dataflow visualization system that provides high flexibility

[3] https://github.com/ICICLE-ai/Smartfoodshed_VA_Flow/tree/main/backend/AutoVega
[4] https://bit.ly/VizKG-MappingRules

Type	Name	Description	Diagram
Data Loader	Tabular Loader	• *Input*: None. • *Output*: List of N objects, $[obj_1, obj_2, ..., obj_N]$. • *Usages*: uploading local CSV files; loading existing CSV files on the Cloud.	*Figure* 4P_3-C
	YAML Loader	• *Input*: None. • *Output*: URL of LinkML. • *Usages*: loading the LinkML of a KG.	*Figure* 4P_1-A
Data Analyzer	Ontology Filter	• *Input*: YAML Loader • *Output*: SPARQL • *Usages*: building the ontology with filters; generating the SPARQL according to user interactions.	*Figure* 4P_1-B'
		Example Query	**Graph Pattern**
		what *(A)* *organization* is located in *(B)* *Great Valley ecoregion*?	
		what *(A)* *program* provides funding for *(B)* *project* to solve *(C)* *wildfire issues*?	
		what *(D)* *wetlands-focused* *(A)* *projects* are lead by *(B)* *organizations* in the *(C)* *Central Coast ecoregion*?	
		who are the *(D)* *owners* of *(B)* *organizations* that work on *(E)* *fragmentation*?	
	Query Editor	• *Input*: SPARQL or None. • *Output*: SPARQL. • *Usages*: viewing/editing queries, i.e., SPARQL	*Figure* 4P_1-D
	KG Qurier	• *Input*: SPARQL • *Output*: List of N objects queried from KG, $[obj_1, ..., obj_N]$ • *Usages*: executing SPARQL to an RDF knowledge graph.	*Figure* 4P_1-C
Data Viewer	Table Viewer	• *Input*: List of N objects. • *Output*: List of M objects (M $\leqslant$ N). • *Usages*: displaying tabular data.	*Figure* 4P_1-E
	Table Visualizer	• *Input*: List of objects. • *Output*: List of objects and charts. • *Usages*: generating Vega-Lite; rendering Vega-Lite specifications to generate images; allowing users to view/edit the Vega-Lite specifications.	*Figure* 4P_2-c1 *Figure* 4P_2-c2

Note: Dashed lines indicate one or more filtering nodes.

Table 1: Our system contains a list of components with pre-defined functions and input/output specifications. Ontology filter contains common graph patterns of $\mathcal{G}'_{ont}$ and example queries of PPOD.

and scalability (**R4**).

User Interface To enable users to build their analysis pipeline flexibly and interactively, the system is designed with two panels: a component list and an edit panel. The component list contains multiple pre-designed components, which are categorized into three groups based on their roles in the analytic process: data loader, analyzer, and viewer. Each component has predefined functionalities and input/output specifications, as indicated in Table 1. Additionally, each component can be minimized or maximized to save space when necessary, e.g., as shown in Figure 4B and B'. In the edit panel, users can add components from the list and drag and drop them to connect them with matched input/output to build their analysis pipeline flexibly.

Data Loader To begin the data analysis pipeline, the first step is typically to load the data. To facilitate this process, our system includes a set of *data loaders* that allow users to specify data sources in various ways and flow the data to subsequent components. One of these *data loaders* is the *tabular loader*, which provides two functions: (1) uploading a local CSV file to the cloud and (2) utilizing an existing file on the cloud. In addition, we have also incorporated feedback from domain experts who have suggested that GitHub is a popular platform for software development collaboration and that it stores many public ontology definitions. To support this, we have included a *YAML loader* that enables users to load expert-defined ontologies through GitHub URLs.

This article has been accepted for publication in IEEE Computer Graphics and Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCG.2023.3263960

Data Analyzer To support the different stages of the data analysis pipeline, we have developed a diverse set of components. These components are designed to perform various tasks, including data processing, querying, and analysis.

Ontology Filters Visualizing the ontology graph G_{ont} to users can help them understand the data structure of knowledge graphs and facilitate interactive visual query building. To achieve this, we introduce *ontology filters*, which takes *YAML loader* and user interactions as input and generates SPARQL queries as output. As shown in Figure 4B', the parsed ontology graph G_{ont} from LinkML is visualized on the right, while a list of dropdown selects corresponding to filtering nodes (●) in G_{ont} is displayed on the left. The user-selected sub-graph G'_{ont} is highlighted in red and triggers the SPARQL generation algorithm.

Query Editors To enhance transparency and interactivity, users are encouraged to review and refine the results of the SPARQL queries generated by *ontology editor*. For this purpose, we introduce *query editors* that can take inputs from either *ontology editor* or user input and output the resulting queries on the editor for use in the subsequent components.

KG Queriers To enable users to query knowledge graphs, we have designed *KG queriers*. Our design logic is based on discussions with domain experts who indicated that public resources are either hosted in the graph database on the Cloud and offer a SPARQL endpoint or stored as static turtle files in GitHub. Therefore, we allow users to specify the source of knowledge graphs by either providing the SPARQL endpoint or a GitHub URL. Additionally, *KG queriers* takes SPARQL queries as another required input to be executed on the specified knowledge graph.

Data Viewer *Table viewers* It is important to provide access to raw data. *Table viewer* presents data in row and column format, allowing users to search and filter the data.

Table visualizers are proposed to render the automatically generated Vega-Lite specifications. It also allows users to save and download the charts as SVG or PNG files. Additionally, they allow users to edit the generated Vega-Lite specification to further refine the visualizations.

System Architecture Figure 1B illustrates the overall architecture of our system. The front-end interface is implemented as a series of web applications using HTML, CSS, and Javascript with the Vue.js framework. The back-end is implemented with Node.js and Flask. When users create the components in the interface, the corresponding API requests are sent to the back-end to execute the algorithm. We deploy our system using the Pods service provided by the Texas Advanced Computer Center (TACC)[5], which provides easy ways to create and monitor the pod's states. The source code and data are available at https://github.com/ICICLE-ai/Smartfoodshed_VA_Flow.

Evaluation of SPARQL generation

We evaluated the performance of our SPARQL generation algorithm by comparing it with SPARQL queries manually created by domain experts. To do this, we invited an expert in semantic web technologies and food ontology to generate different types of queries, each with their corresponding SPARQL. The queries were based on the four types of schemes we described. The expert used the Gruff AllegroGraph interface to build the SPARQL queries of PPOD, denoted as q'. We then compared the querying results separately for our generated queries q and q'. We found that our q achieved the same querying performance as q'. As an example for qualitative comparison, we present q and q' of the query *"Which non-profit organizations work on water quality in Yolo or Solano counties?"* The expert made query q' is shown as follows:

```
SELECT DISTINCT ?node_variable_1
WHERE {
VALUES ?county {
<http://www.wikidata.org/entity/Q109709>
<http://www.wikidata.org/entity/Q108083> }
?node_variable_1 rdf:type foaf:Organization ;
<http://www.w3.org/ns/org#classification>
<https://raw.githubusercontent.com/adhollander/
    FSLschemas/main/CA_PPODterms.ttl#oty_cf5070> ;
<https://raw.githubusercontent.com/adhollander
    /FSLschemas/main/fsisupp.owl#inCounty> ?county ;
<https://raw.githubusercontent.com/adhollander
    /FSLschemas/main/fsisupp.owl#FSI_000239>
<https://raw.githubusercontent.com/adhollander
    /FSLschemas/main/sustsourceindiv.rdf#CI0303> .}
```

Our algorithm generates the following SPARQL queries:

```
PREFIX dcterms: http://purl.org/dc/terms/
PREFIX core: http://vivoweb.org/ontology/core#
```

[5] https://tapis-project.github.io/live-docs/?service=Pods

This article has been accepted for publication in IEEE Computer Graphics and Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCG.2023.3263960

```
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#
PREFIX fsls: https://raw.githubusercontent.com
    /adhollander/FSLschemas/main/fsisupp.owl#
PREFIX fsli: https://raw.githubusercontent.com
    /adhollander/FSLschemas/main/sustsourceindiv.rdf#
SELECT *
WHERE {
?organization dcterms:title ?organization_title .
?organization rdfs:label ?organization_label .
?organization a foaf:Organization .
?organization fsls:FSI_000239 ?componentissue .
Filter (?componentissue IN(fsli:CI0303) ) .
    ?organization fsls:inCounty ?county .
Filter (?county
    IN(<http://www.wikidata.org/entity/Q108083>,
    <http://www.wikidata.org/entity/Q109709>) ) .
?organization org:classification ?organizationtype .
Filter (?organizationtype IN(fslp:oty_cf5070) ) .
}
```

It is clear to see we use prefixes to replace long label names, which can make queries more readable, especially when the same prefix is used multiple times in the WHERE clause.

Evaluation of IKLE

We have conducted a systematic comparison between our system and the seven most related systems that we have introduced in the related works section.

To ensure the comparison is comprehensive and convincing, we have carefully designed five tasks to test each system, as shown in Figure 3. (1) *Information display* evaluates the information display capability of each system. This task checks whether the system can efficiently present information with sufficient details. (2) *Data visualization & graph visualization* assesses whether the system supports effective data visualization. It involves mapping data to visual channels and providing recommendations for data visualization to help users understand the data more easily. (3) *Graph exploration & efficient query* examines whether the system supports efficient graph query. It includes providing guidance on generating queries automatically based on user interaction and enabling users to fine-tune queries interactively. (4) *Flexible IR pipeline* evaluates whether the system supports users in complex information-retrieving tasks with a customized retrieval pipeline. (5) *Gain insights* tests the system's ability to generate insights that assist users in solving real-world tasks.

The comparison is shown in Figure 3. In conclusion, our systematic comparison of IKLE with seven other existing visual analytics systems has shown that IKLE not only meets but also exceeds the requirements of the five tasks we designed. Our system efficiently presents information with sufficient details, supports effective data visualization, facilitates efficient graph visualization and query, supports complex graph information retrieval tasks with a flexible retrieval pipeline, and provides great insight finder ability. We are confident that our findings demonstrate the unique strengths of IKLE and its potential as a valuable tool for knowledge graph exploration and analysis.

Case Study

To demonstrate the usefulness of our system in smart foodsheds, we present the results of applying our system in real-world scenarios.

Finding partners in smart foodsheds.

Alice wants to identify potential collaborators who are leaders on water pollution issues in the food system. She uses our system to search the PPOD knowledge graph for relevant information. The analysis pipeline is shown in Figure 4P_1. She began by using a *YAML loader* (Figure 4P_1-A) to load the LinkML files of PPOD. She then connected it to an *ontology filter* (Figure 4P_1-B’) to construct queries. In the *ontology filter*, she selected the related filtering conditions on the left dropdown list, including “wastes & pollution”, “water” under the *integrated issues* (Figure 4P_1-b_2) and “advisor,” “board member,” “director,” and “elected official” under the *position type* (Figure 4P_1-b_1). She then enabled the lasso function and brushed a subgraph as G_{ont} on the right panel (Figure 4P_1-b_3). Once the G_{ont} is changed, the output of the *ontology filter* will be updated accordingly.

Alice trusted our automatic generation. Without fine-tuning it, she connected the *ontology filter* directly to a *KG querier* to perform the query. She added a *table viewer* at the end of the pipeline to display the results. When the SPARQL execution is finished in the *KG querier*, the *table viewer* is automatically updated with the results (Figure 4P_1-E), allowing Alice to find possible partners for collaboration. This shows that our system can be used for partner finding and information retrieval, which are important for the effective management of food systems.

This article has been accepted for publication in IEEE Computer Graphics and Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCG.2023.3263960

-	VisFlow	VisComposer	VizKG	Neo4j	GraphPolaris	VisQL	AllegroGraph	**IKLE**
Information Display	●	●	●	◑	●	●	●	●
Data Visualiztion & Graph Visualization	●	◑	●	◑	●	◑	●	●
Graph Exploration & Efficient Query	◑	◑	◑	◑	●	○	●	●
Flexible IR Pipeline	●	●	○	○	○	○	○	●
Gain Insights	◑	◑	○	○	◑	○	◑	●

Figure 3: Performance evaluation between IKLE with seven related systems for knowledge graph.

Information Retrieval

We present a case study to demonstrate the flexibility and scalability of our system in retrieving information from other knowledge bases. Jason, a Ph.D. student in food science, uses our system to search for the latest research in food literature using SPARQL queries. He has some familiarity with RDF and SPARQL.

First, Jason dragged a *query editor* and wrote a SPARQL query to identify study fields related to food (Figure 4P_2-A). He then connected to a *KG querier* and input the endpoint of the Microsoft Academic Knowledge Graph (MAKG) as the knowledge base (Figure 4P_2-B). He joined a *table visualizer* to explore the results. As visualized in Figure 4P_2-c_1, Jason found that the citations have decreased since 2018 but started to increase in 2020. He also saw that the *health food* field had gained more attention since 2020, possibly due to the influence of the pandemic on people's work-life balance.

The *table visualizer* displayed the distribution of publication numbers in subfields (Figure 4P_2-c_2), allowing Jason to see the most popular fields were *food preservation, functional food, and fermentation in food processing*. This case study demonstrates that our system can retrieve information from various knowledge bases and can be customized to meet user needs.

Resilience Analysis

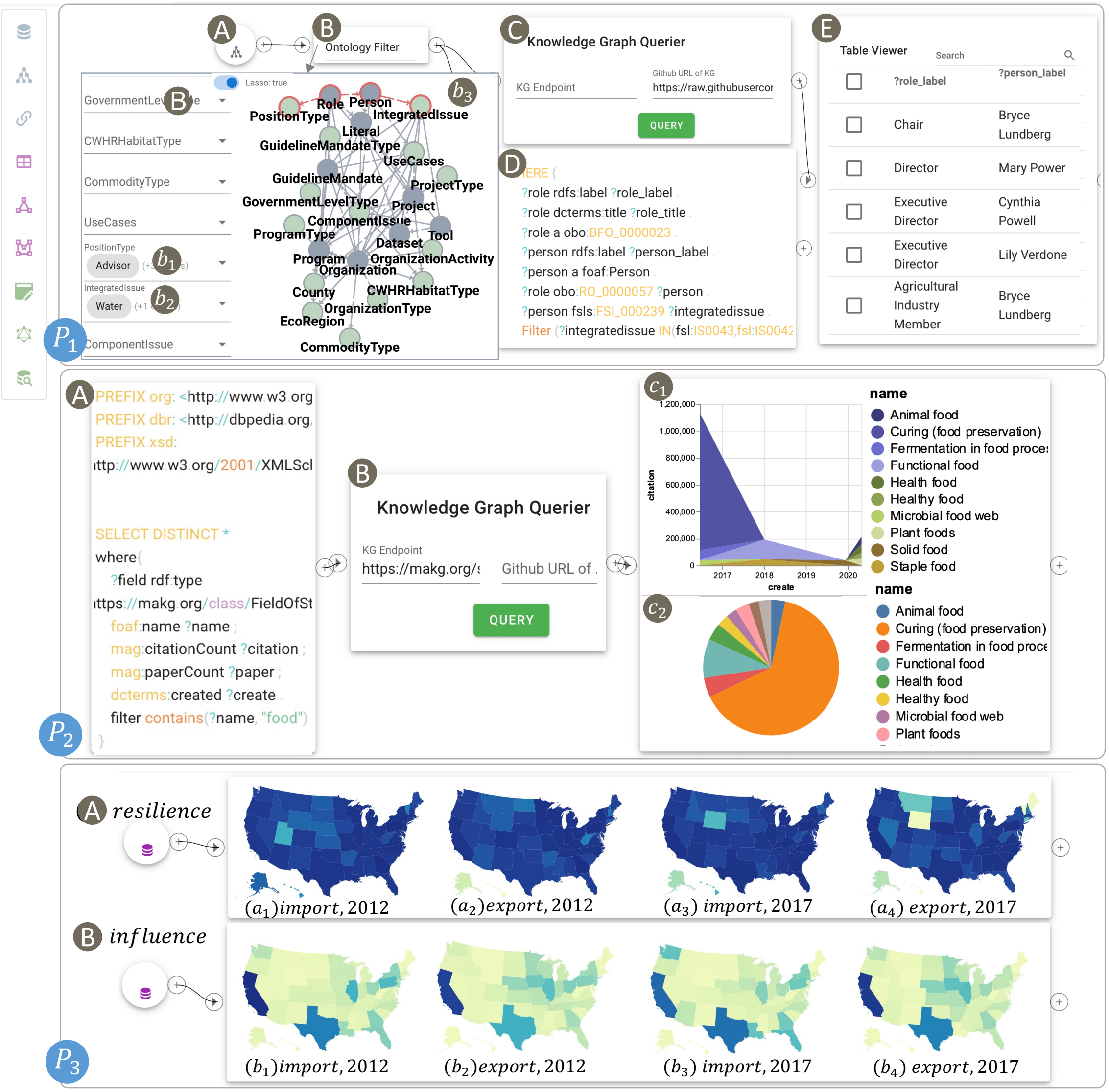
Due to the high flexibility of our system, users can also load the tabular data and visualize the data patterns. Quantifying the resilience of the food flow network is an important task in the food system to identify potential security issues. Our expert, Jasper, proposed a novel way to measure the resilience of the US multi-commodity flow network [20]. He wants to use our system to explore computed resilience scores to gain valuable insights.

The constructed analysis pipeline is visualized in Figure 4P_3. Jasper first computed resilience for each state (node) in the agricultural multi-commodity flow network in 2012 and 2017 and loaded it through a *data loader*. He then connected it to a *data visualizer*. From the node-level resilience results (Figure 4P_3-A), Jasper observed that the states along the east coast, west coast, and midwest have higher resilience than states in other geographic regions. He inferred that this was because these states had a mid-range flow of products - not too big or too small, making them more resilient to disruption risks. This alerted him to the potential brittleness of supply chains that relied heavily on some states and the lack of food access in others.

Similarly, he calculated the import/export influence of each state on the entire food flow spatial network in 2012 and 2017. As shown in Figure 4P_3-B, he identified that many top agricultural-producing states had import/export influence on the network, such as Texas, California, and the mid-west states. He could see the importance of these states to North American and global supply chains and that geographic concentration increased from 2012-2017. This led Jasper to ask further questions about regional differences in crops produced and markets served and the relationship these states had to ports for import/export markets.

Expert Feedback

We conducted free-form interviews with domain experts to gather their feedback on the system, asking them about their likes and dislikes and for any suggestions they had. Overall, they agreed that the system makes it easier for users to



```
ERE {
  ?role rdfs:label ?role_label .
  ?role dcterms:title ?role_title .
  ?role a obo:BFO_0000023 .
  ?person rdfs:label ?person_label .
  ?person a foaf:Person .
  ?role obo:RO_0000057 ?person .
  ?person fsls:FSI_000239 ?integratedissue .
  Filter (?integratedissue IN(fsl:IS0043,fsl:IS0042
```

	?role_label	?person_label
□	Chair	Bryce Lundberg
□	Director	Mary Power
□	Executive Director	Cynthia Powell
□	Executive Director	Lily Verdone
□	Agricultural Industry Member	Bryce Lundberg

```
PREFIX org: <http://www.w3.org
PREFIX dbr: <http://dbpedia.org,
PREFIX xsd:
ıttp://www.w3.org/2001/XMLScl


SELECT DISTINCT *
where{
   ?field rdf:type
ıttps://makg.org/class/FieldOfSt
   foaf:name ?name ;
   mag:citationCount ?citation ;
   mag:paperCount ?paper ;
   dcterms:created ?create .
   filter contains(?name, "food")
}
```

Figure 4: Three pipelines in case studies. (P_1): finding partners in smart foodsheds using PPOD. (A) *YAML loader* to load LinkML of PPOD; (B) minimized view and (B') maximized view of *ontology filter*; (b_1, b_2) user-selected filters; (b_3) user-brushed query graph G'_{ont}; (C) *KG querier* to execute SPARQL on PPOD; (D) *code editor* to display the generated SPARQL; (E) *table viewer* displays the queried data. (P_2): Information retrieval from Microsoft Academic Knowledge Graph. (A) *query editor* to write the SPARQL; (B) *KG querier* specified with the MAKG endpoint; (c_1, c_2) *table visualizer* to reveal the data patterns from results. (P_3): State-level resilience and influence in the US agricultural multi-commodity flow network in 2012 and 2017.

complete complex tasks and achieve their goals efficiently. E1 noted that the system has great potential to *democratize access to data visualization and promote the co-production of knowledge.* They also highlight the system's ability to incorporate knowledge graphs and tabular data, which is particularly *useful for small and medium-scale practitioners, like independent grocers.*E1 also mentioned that the system is valuable for scientists who may not be familiar with AI, as it provides access to high-quality visualizations of data patterns. In the PPOD case study, the system demonstrated its potential by *showing the value of developing a food systems ontology foundry for aligning databases and facilitating federated learning.* Another expert E2 expressed excitement about the system's potential to benefit a wide range of users and looked forward to its continued development and success. E3 expressed interest in using the system to explicitly map a wide range of sustainability activities taking place in working landscapes, potentially *providing valuable insights into how to better manage and sustain these landscapes.*

Conclusion

In this paper, we introduce an interactive knowledge and learning environment (IKLE) to support various interactions with knowledge graphs in smart foodsheds. To establish this intelligent environment, we collaborated with domain experts to design a novel dataflow system. Furthermore, the three case studies manifest their usefulness to policymakers and practitioners working in smart foodsheds under various scenarios. It also demonstrates that our system can be easily extended to other domains, making it a useful tool for a wide range of applications.

Acknowledgements

This work is financially supported by the NSF-funded AI institute [Grant No. OAC-2112606].

■ REFERENCES

1. Deagen, Michael E., et al. "FAIR and Interactive Data Graphics from a Scientific Knowledge Graph." Scientific Data 9.1 (2022): 1-11.
2. Auer, Sören, et al. "Dbpedia: A nucleus for a web of open data." The semantic web. Springer, Berlin, Heidelberg, 2007. 722-735.
3. Vrandečić, Denny, et al. "Wikidata: a free collaborative knowledgebase." Communications of the ACM 57.10 (2014): 78-85.
4. Holden, Nicholas M., et al. "Review of the sustainability of food systems and transition using the Internet of Food." npj Science of Food 2.1 (2018): 1-7.
5. Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
6. Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.
7. Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).
8. Stolte, Chris, et al. "Polaris: A system for query, analysis, and visualization of multidimensional relational databases." IEEE Transactions on Visualization and Computer Graphics 8.1 (2002): 52-65.
9. Hanrahan, Pat. "Vizql: a language for query, analysis and visualization." Proceedings of the 2006 ACM SIGMOD international conference on Management of data. 2006.
10. Waser, Jurgen, et al. "Nodes on ropes: A comprehensive data and control flow for steering ensemble simulations." IEEE transactions on visualization and computer graphics 17.12 (2011): 1872-1881.
11. Yu, Bowen, et al. "VisFlow-Web-based visualization framework for tabular data with a subset flow model." IEEE transactions on visualization and computer graphics 23.1 (2016): 251-260.
12. Yu, Bowen, et al. "FlowSense: A natural language interface for visual data exploration within a dataflow system." IEEE transactions on visualization and computer graphics 26.1 (2019): 1-11.
13. Javed, Waqas, et al. "ExPlates: spatializing interactive analysis to scaffold visual exploration." Computer Graphics Forum. Vol. 32. No. 3pt4. Oxford, UK: Blackwell Publishing Ltd, 2013.
14. Mei, Honghui, et al. "Viscomposer: A visual programmable composition environment for information visualization." Visual Informatics 2.1 (2018): 71-81.
15. Xiang, Xiayu, et al. "Knowledge graph-based clinical decision support system reasoning: a survey." 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC). IEEE, 2019.
16. Shao, B., et al. "A survey of research hotspots and frontier trends of recommendation systems from the

This article has been accepted for publication in IEEE Computer Graphics and Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCG.2023.3263960

perspective of knowledge graph." Expert Systems with Applications, 165, 113764 (2021).

17. Hollander, Allan D., et al. "Toward smart foodsheds: Using stakeholder engagement to improve informatics frameworks for regional food systems." Annals of the American Association of Geographers 110.2 (2020): 535-546.
18. Chen, Yu, et al. "Personalized food recommendation as constrained question answering over a large-scale food knowledge graph." Proceedings of the 14th ACM International Conference on Web Search and Data Mining. 2021.
19. Raissya, Hana, Fariz Darari, and Fajar J. Ekaputra. "VizKG: A Framework for Visualizing SPARQL Query Results over Knowledge Graphs." VOILA (2021).
20. Rao, Jinmeng, et al. "Measuring network resilience via geospatial knowledge graph: a case study of the us multi-commodity flow network." Proceedings of the 1st ACM SIGSPATIAL International Workshop on Geospatial Knowledge Graphs. 2022.
21. IBM SPSS Modeler. http://www.ibm.com/software/products/en/spss-modeler.

Yamei Tu is a Ph.D. student at OSU, and her research interests are visualization and NLP. Contact her at tu.253@osu.edu.

Xiaoqi Wang is a Ph.D. student at OSU, and her research interests include explainable AI and graph drawing. Contact her at wang.5502@osu.edu.

Rui Qiu is currently a Ph.D. student at OSU and his research interests include visual analysis and clinical informatics. Contact him at qiu.580@osu.edu.

Han-Wei Shen is a full professor at the Ohio State University. His primary research interests include scientific visualization and computer graphics. Contact him at shen.94@osu.edu.

Michelle Miller works as a practicing economic anthropologist engaged in participatory action research. Contact her at mmmille6@wisc.edu.

Jinmeng Rao is a Ph.D. student at the University of Wisconsin-Madison. Contact him at jinmeng.rao@wisc.edu.

Song Gao is an Associate Professor at the University of Wisconsin-Madison and the director of the Geospatial Data Science Lab. Contact him at song.gao@wisc.edu.

Patrick R Huber is a Project Scientist with the Food Systems Lab. He directs the working landscapes portfolio of projects for FSL. Contact him at prhuber@ucdavis.edu.

Allan D Hollander is a geographer whose work focuses on the use of information systems for environmental management. Contact him at adhollander@ucdavis.edu.

Matthew Lange is the CEO and CSO of the International Center for Food Ontology Operability Data and Semantics (IC-FOODS). Contact him at matthew@ic-foods.org.

Christian R Garcia is currently working in CIC on Abaco. Contact him at cgarcia@tacc.utexas.edu.

Joe Stubbs leads the Cloud and Interactive Computing (CIC) group. Contact him at jstubbs@tacc.utexas.edu.