City, University of London

MSc in Data Science

Project Report

2020-2021

Domain Knowledge Graph Visualisation: Food Waste Policy Case Study

Oliver Keers

Supervised by: Dr Radu Jianu

Submitted: 1st October 2021

Declaration

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

Signed:

Abstract

The food system is a dynamic global network of numerous, diverse entities with complex interrelationships. It is not modelled comprehensively which makes it difficult to understand. Food policies are actions within the food system that aim to change or otherwise regulate it. Developing a visualised, accessible model of the food system has potential to help improve the work of policymakers, support the learning of researchers, and improve corporate compliance. This work designs, builds & evaluates such a system: a domain knowledge graph visualisation using food waste policy as a case study.

A food waste ontology, domain knowledge graph and interactive visualisation are developed using a novel parallel approach in collaboration with domain experts. These are evaluated using visual analytics, use-case completion, and user study involving five other independent domain experts. The visualisation innovates over existing tools by tailoring its design to a specific knowledge graph, and a non-technical domain-expert userbase. During evaluation we found that the software supported domain experts in reflection and reasoning, who thought it would be of use in their work. The significance of these results is discussed, and we suggest approaches for future work in domain knowledge graph visualisation.

Contents

1 Introduction & Objectives

The food system is a dynamic global network consisting of numerous, diverse entities with complex interrelationships. It is not modelled comprehensively which makes it difficult to understand. Food policies are actions within the food system that aim to change or otherwise regulate it. Developing a visualised, accessible model of the food system has potential to help improve the work of policymakers, support the learning of researchers, and improve corporate compliance. This work designs, builds & evaluates such a system: a domain knowledge graph visualisation using food waste policy as a case study.

1.1 Background to the problem

Modelling and visualising the complex and diverse network of policies governing food would support practitioners within the domain. Food policy does not exist as a single, well-defined entity. There are many pieces of legislation, drawn up over time by different organisations, affecting different aspects of the food system. This abstract network is very important, with effects beyond politics including public health, the economy, and society at large. However, no consolidated source of information on food policies exists, and nor has the network of relationships been codified. This makes even expert understanding subjective and incomplete. This problem could be addressed by developing a computational model of this system and visualising it for food policy experts to use.

This problem is suitable for modelling with a domain knowledge graph. A domain knowledge graph aims to semantically model the vast network of directed relationships between entities within that specific domain(Paulheim, 2016; Abu-Salih, 2021; Hogan *et al.*, 2021). Knowledge graphs build on ontologies that develop domain-specific hierarchies of terms. Various classes of entities and relationships can be modelled. Information is consolidated from diverse sources into a single resource. Food policy networks are vast abstract networks of multiple classes of entity, with a range of relationships, and information stored in various sources. This makes domain knowledge graphs well-suited for modelling this system.

Visualisation of knowledge graphs improves their accessibility, and can support human reasoning, research, and reflection. Knowledge graphs are abstract conceptual networks of relationships stored as text. Knowledge graphs can be thought of as like a relational database, with humans able to access information through a query language (SPARQL). Food policy experts do not typically possess the programming skills required to query knowledge graphs. Visualisation removes this barrier. Visualisation also facilitates human reasoning about data and helps analysts to develop their domain understanding (Endert *et al.*, 2014; Andrienko *et al.*, 2018). Thus, visualisation would allow non-technical users, such as food policy experts, to use a knowledge graph to reason more effectively about a domain.

1.2 Research Question & Objectives

The project aimed to answer the following research question:

RQ: How can a domain knowledge graph of food policy be visualised effectively to support the reflective and reasoning process of domain experts?

The project outcomes were achieved in pursuit of the following research objectives:

RO1: To develop an ontology in collaboration with food policy experts and using this to construct and refine a knowledge graph of the domain.

RO2: To design, implement and evaluate interactive knowledge graph visualisations that effectively support the reflective and reasoning process of domain expert users, and can be understood by non-technical audiences.

1.3 Outline of Methods

This work is a case study, using Food Waste, to develop & evaluate an ontology, knowledge graph, and knowledge graph visualisation. The ontology was constructed through domain expert observation and interview. The knowledge graph was constructed by iterating through manually encoded datasets on food waste. The knowledge graph visualisation was developed using iterative prototyping in conjunction with domain experts. The resulting knowledge graph visualisations were evaluated through user study on other food policy experts.

1.4 Project Beneficiaries & Major Contributions

This work directly benefits knowledge graph developers, visualisation practitioners, and food policy experts, with indirect benefits to society at large. A wider domain-expert userbase can better provide feedback on knowledge graphs, allowing for refinement and improvement. This in turn increases the utility of knowledge graphs. The visualisation approaches presented can be of use for developing visualisations of other domain knowledge graphs. Food policy experts benefit through developing new perspectives on their domain, without needing to learn new technologies. These perspectives can help the domain experts in developing better food policies, providing benefits to society at large.

This work makes the following contributions:

- An ontology modelling the major classes and interactions within food waste, generalisable beyond the domain.
- A small domain knowledge graph of individual instances of these classes and interactions
- Interactive visualisation software that allows non-technical users to reason about food waste using the knowledge graph.
- A novel parallel approach to development of knowledge graphs and their visualisations, with visualisation as the primary objective.
- Evaluation of both the software and its development, with principles that can be generalised beyond this domain.

1.5 Major Changes from Proposal

This work focussed more heavily on visualisation rather than knowledge graph construction as originally proposed[1]. This work was conceived as the second project of a set of three, with the first project extracting a dataset to be used in this work. This did not happen within the timeframe of the project, and so greater weight was placed on visualisation aspects. Nevertheless, the initial work plan was retained.

[1] Appendix A

2 Context

Information about food policies is complex and fragmented. A knowledge graph of food policy could consolidate this and would be a valuable resource for practitioners. Currently no such domain knowledge graph exists.

Using knowledge graphs requires technical knowledge not typically held by food policy experts. Visualising these knowledge graphs has the potential to improve accessibility. No existing visualisations of knowledge graphs are designed for non-technical domain expert audiences.

Development of a knowledge graph in parallel with its visualisation, and prioritising the latter, is novel. This approach has the potential to improve human interpretability.

Visualisation of small, domain knowledge graphs is uncommon but allows for tailored interactions and features not applicable when visualising larger, general knowledge graphs.

2.1 Domain Knowledge Graph

A knowledge graph of food policy does not exist but is needed to capture the semantics of the landscape.

Food policy is fragmented and understanding the abstract networks present in food systems is a current research challenge. Sixteen government departments have involvement in food policies in the UK(Parsons, 2020). There is no single source of information on food policies, making research challenging. The complexity of food systems is acknowledged(Ericksen, 2008), with even small local groups involved in food policy warranting network analysis(Levkoe *et al.*, 2021). One current avenue of research on food policies involves modelling and visualising small food systems to understand them, typically through causal loop diagrams (Boelsen-Robinson *et al.*, 2021). Such work includes modelling dairy food systems in Nairobi (Kiambi *et al.*, 2018) or corporate political activity in South Africa (Mialon, Crosbie and Sacks, 2020). However, this is not performed with a formal ontology to provide a structure, so entities and relationships are not modelled consistently. Further, construction of these diagrams is usually performed manually, limiting their scope and comprehensiveness. Where efforts have been made to understand global systems (Savary *et al.*, 2020) or visualise food systems data using dashboards(Fanzo *et al.*, 2020), the detail of relationships is lost.

Many knowledge graphs and ontologies of food exist, with several notable examples focussed on ingredients and recipes. Batista *et al.* (2006) detail the full development process for an early cooking ontology, which was later made accessible to users through the development of a dialogue system (Martins *et al.*, 2008). FRUCT (Kolchin and Zamula, 2013) is a relatively simple ontology describing the ingredients and constituents of foods. FoodON (Dooley *et al.*, 2018) is a more comprehensive ontology incorporating details such as biological taxonomy, processing methods and places of origin. Foodbar (Zulaika, Gutiérrez and López-de-Ipiña, 2018) is a knowledge graph used to recommend tapas recipes based on user inputs. FoodKG (Haussmann *et al.*, 2019) is another knowledge graph designed to recommend recipes based upon various criteria. Incorporating FoodON and other ontologies, users can execute SPARQL queries on FoodKG to find recipes meeting criteria such as ‘a quick-to-make spicy beef and tomato dish, suitable for coeliacs”. Recent work (Shirai *et al.*, 2021) suggests ingredient substitutions for recipes, based on cosine similarities of vectors derived from this knowledge graph.

Other knowledge graphs of food relate to its production and agriculture, rather than governance. AGROVOC (Caracciolo *et al.*, 2013) is a dictionary of terms produced by the Food and Agriculture Organisation of the United nations, covering its work and linking to equivalent thesauri from different organisations, in different languages, facilitating interoperability. CAVOC (Takezaki *et al.*, 2020) is a recent example of an ontology that builds upon AGROVOC. CAVOC aims to align Japanese agricultural terminology with international semantic standards. The need for this interoperability is illustrated in the work of Qin, Hao and Zhao (2019), whose work to develop a food safety knowledge graph was limited by an absence of work on food ontologies in Chinese. AgriKG (Chen *et al.*, 2019) demonstrates how a knowledge graph of agriculture can be useful in practice, integrating question answering and search-by-image functionality.

This work makes novel contributions in developing an ontology and knowledge graph of food waste policy. As far as we are aware, there is no knowledge graph of food policies, regulations, and governance. Developing a knowledge graph of all policies relating to food systems is a substantial task, outside the scope of this project. This project uses food waste policy as a case study, illustrating the potential of a larger knowledge graph to advance work within the domain.

2.2 Knowledge Graph Visualisation

Effective visualisation of knowledge graphs has the potential to improve their accessibility, particularly for users with no semantic web experience.

Visualisation helps novice users to access knowledge graphs, which are otherwise difficult to use. Knowledge graphs exist as vast networks of linked data, similar to a relational database. Interfacing with them usually requires use of SPARQL, a specialist query language similar to SQL that is not used outside of semantic web contexts. Effective visualisation of ontologies (and thus knowledge graphs) can improve accessibility, with node-link diagrams shown to be particularly effective at supporting novice understanding (Asmat, Wiens and Lohmann, 2018).

Many visualisation tools for large knowledge graphs exist, with no consistent approach to their design. Different knowledge graph and ontology visualisations are aimed at very different audiences and have vastly different design and functionality(Po and Papastefanatos, 2020; Po *et al.*, 2020). A very brief overview of an illustrative sample of tools is provided below.

"Browser-style":

- LODview (Bellini, Nesi and Venturi, 2014) represents each URI as a webpage with a series of navigable links to connected pages, with users manually entering a valid start point to begin browsing. This provides a very detailed view of each URI, but it is difficult to understand the network of relationships between multiple pages.
- Balloon Synopsis (Schlegel *et al.*, 2014) is another browser-style interface with a page per node. This is more heavily stylised, with each linked entity represented as a 'tile' containing its name and relationships. Users can navigate to new nodes via these tiles.
- RDF Surveyor (Vega-Gorgojo *et al.*, 2019) is a more recent implementation of a similar, browser-like visualisation. Users can browse URIs contained within SPARQL endpoints as a series of hyperlinked webpages. Links are given in tabular format with source/destination and endpoint. Again, this provides a very detailed view of a particular instance, but it is difficult to understand graph structure.

Aggregation:

- LD-VizWiz (Atemezing and Troncy, no date) provides summary information about SPARQL endpoints. Limited graphical representations of the data are available (such as geographical or statistical data), but this is largely for aggregate reporting purposes.

Node-link (entity-level):

- RelFinder (Lohmann *et al.*, 2010) provides a node-link representation of a knowledge graph, focussed on finding the links between user-defined entities. These links can be indirect, via intermediate nodes. Filtering options enable users to focus on things of interest to them. Nodes are used to display edge labels, which is visually disconcerting.
- LodLive (Camarda, Mazzini and Antonuccio, 2012), offers a node-link representation, expanded through user interactions from a user-defined start point. The approach to iterative expansion allows users to explore area of interest in increasing depth. Layout is simple (nodes add radially), which causes the display to become cluttered after adding only a handful of nodes.
- LODmilla (Micsik, Turbucz and Tóth, 2015) is a second node-link visualisation, allowing for node-search and iterative expansion of two pre-defined endpoints. This is a relatively novice-friendly visualisation tool with editing capabilities. Unlike LodLive, layout options exist, although similar problems with clutter exist.

Node-link (schema-level):

- H-BOLD (Po and Malvezzi, no date) is a further node-link diagram, allowing access to a number of endpoints. Where H-BOLD differs from the other representations is on its level of abstraction. H-BOLD does not represent instances within the dataset, only showing information at the class-schema level. This is useful for understanding the structure of large knowledge bases, but provides no information on individuals.
- LD-VOWL (Weise, Lohmann and Haag, 2016) is similar in offering a schema-level node-link view of knowledge graphs. This has a simpler interface then H-BOLD, making it more usable for novices, although it accordingly has much less functionality.

Node-link (3D)

- Tarsier (Viola *et al.*, 2018) is unique in being a 3D node-link visualisation tool, which is designed for semantic web students and developers and this functionality is mirrored in its complexity. The additional dimension is used to move instances corresponding to a query, without users needing to know SPARQL. These planes then support human inference based on the connections between the two.

"Visualised Querying":

- Sparklis (Ferré, 2014), provides support in writing and executing SPARQL queries on a selected endpoint and visualising their outputs. The outputs are visualised in tabular format, as it focusses on information retrieval rather than the connections between it.
- QueryVOWL (Haag *et al.*, 2015) provides a visual interface for constructing complex SPARQL queries. Users input a starting point, and construct the query using a node-link visualisation to add complexity. They are able to retrieve the results in tabular format, or extract the query as written in SPARQL.
- OptiqueVQS (Soylu *et al.*, 2018) provides a very rich visual interface for constructing queries of domain-specific ontologies. It is also interested in information retrieval, rather than forming connections, and so outputs are tabular, with links.

Specialist:

- Protégé (Musen, 2015) is at the most technical end of the spectrum. Protégé is aimed at semantic web specialists, representing ontologies or knowledge graphs as a nested hierarchy. Protégé in effect offers a graphical user interface and several tools to create and maintain ontologies and knowledge graphs.

This work is unusual in aiming its visualisation at users who have no experience in programming or semantic web technologies. All the above tools require at least an awareness of knowledge graphs to use effectively. It is broadly true that the tools with increasing expressiveness and interactivity come with increased complexity, and a higher barrier to entry. This work aims to offer rich interactivity, representing the complexity of a knowledge graph in an accessible manner.

2.3 Prioritising Visualisation

This work is unusual in developing a knowledge graph in parallel with its visualisation and prioritising the latter.

Knowledge graphs are primarily developed to structure data within computers and facilitate intelligent computation – human use is a secondary goal. Knowledge graphs are often vast - manual inspection of their entirety is usually not feasible (Paulheim, 2016). Of the 69 metrics for assessing the quality of linked data (Zaveri *et al.*, 2015), only two of these relate to human interpretation. Some knowledge graphs, like Wikidata (Vrandečić and Krötzsch, 2014), do not use human readable URIs. These systems have not been designed for direct human interaction.

This work is unusual in its use of a knowledge graph primarily to facilitate human reasoning. Visualisation allows people to understand data through building their mental model of the domain(Andrienko *et al.*, 2018). Use of a programmatic reasoner to check for logically unsatisfiable statements in a knowledge graph is standard practice (Zaveri *et al.*, 2015; Paulheim, 2016). However, it is not common to design a knowledge graph to support the reasoning of a human analyst, a practice from visual analytics (Endert *et al.*, 2014).

Almost all knowledge graph visualisation tools were designed a posteriori for pre-existing knowledge graphs, which may compromise the effectiveness of the visualisation. The only notable exception to this is Protégé which is more of an ontology creation/editing interface than a visualisation tool. Otherwise, visualisations are typically developed to add to an existing graph(Po *et al.*, 2020). The visualisations aim to represent a system designed for machines, including aspects that may confuse humans. A simple example of such an issue is the disjoint relationship, as visualised in Fig. 1 using WebVOWL (Lohmann *et al.*, 2015). This is important for a typical knowledge graph as it represents that there is no overlap between two classes. When a disjoint relationship visualised it appears to indicate a connection between them. For a human, simply seeing two non-overlapping nodes is a more effective indication of their disjointedness. Modelling this disjoint relationship in the knowledge graph becomes detrimental to the visualisation.

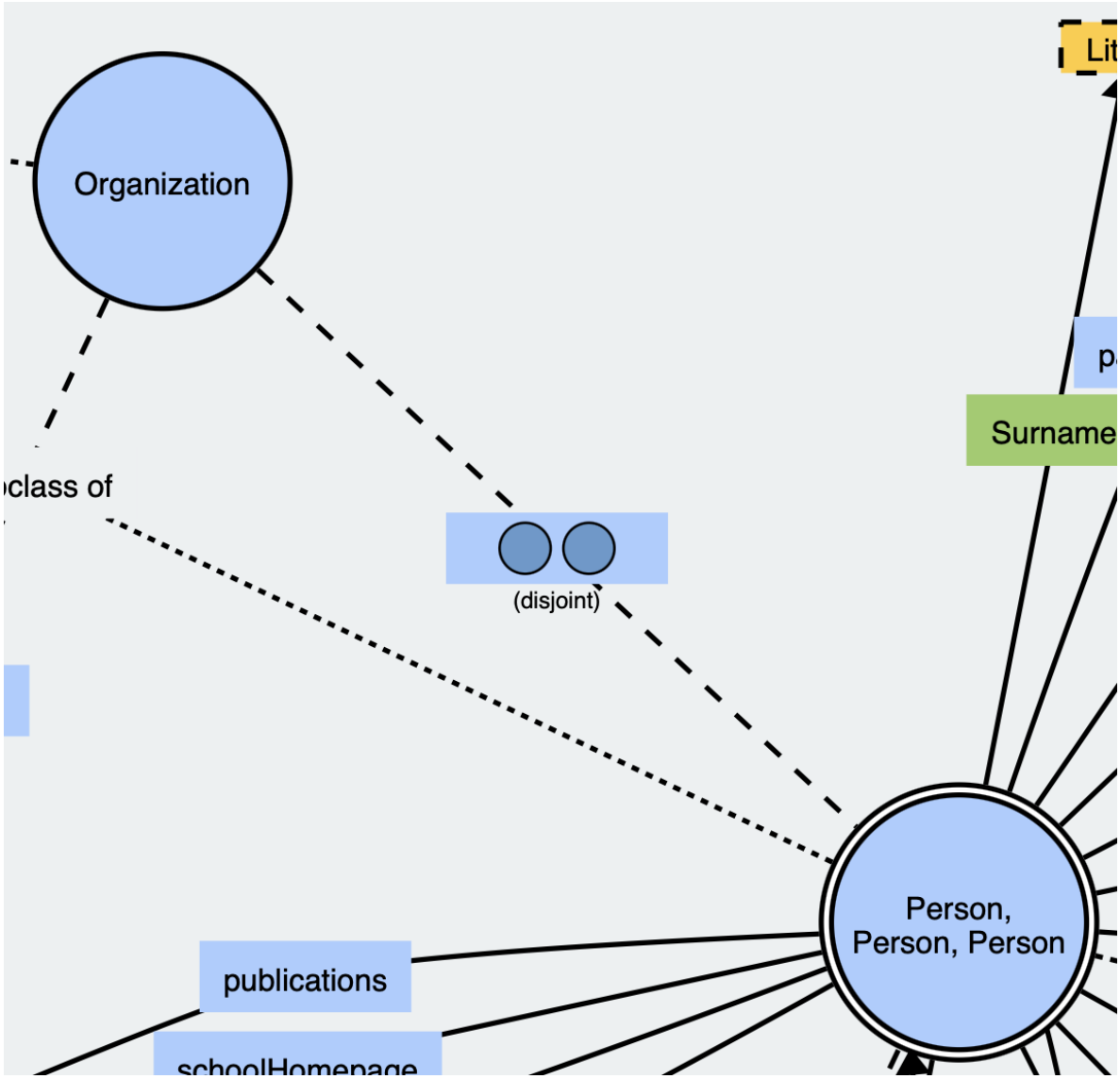


Fig. 1 – A visualisation of the FOAF ontology using WebVOWL, illustrating the disjoint relationshiup

This work is novel in having effective visualisation as its primary goal and designing the knowledge graph synergistically with the visualisation to facilitate this. Usually, developers produce a visualisation ***of*** a knowledge graph. Here, we build a knowledge graph ***for*** visualisation. This subtle change in priorities significantly affects knowledge graph design choices. Human interpretability of URIs becomes highly important, while concise URIs are far less so. Linking to external knowledge graphs ceases to be relevant. The knowledge graph becomes a tool to support the visualisation

2.4 Visualising Small Domain Knowledge Graphs

This work focuses on visualizing a small domain knowledge graph, which allows for tailored interactions and features not applicable when visualising larger, general knowledge graphs.

Designing a visualisation for a small knowledge graph allows for the implementation of functionality and optimisations not possible or desirable in most tools for larger graphs. All the tools outlined above are built with a view to visualising very large knowledge graphs. This is appropriate, given the typical scale of a knowledge graph. Large graph visualisations require specific approaches to process their complexity and scope(Chawuthai and Takeda, 2016). This means that functionality that would be desirable for a small dataset, such as overviews, may absent. A recent evaluation of visualisation tools found that only one, Sparklis, allowed users to view all instances (Po and Papastefanatos, 2020). Sparklis offers this through a tabular, paginated view and this provides no sense of structure. For a dataset like DBPedia, visualising 9.5 billion facts is unlikely to be of any use. With a substantially smaller knowledge graph, an overview of all data could be a useful starting point for exploration and may even provide insights into the graph. For large knowledge graph, data is retrieved dynamically via connection to a SPARQL endpoint, which is a highly latent process. Results may also be incomplete, with endpoints limiting result set sizes. Designing a visualisation for a smaller knowledge graph allows for optimisations such as caching or precomputation of layout.

This work is novel in developing a small knowledge graph visualisation with food policy-specific interactions. Most knowledge graph visualization tools are designed for generalised knowledge graphs and do not offer domain-specific interactions. Of the visualisation tools outlined previously, only OptiqueVQS was developed with a specific domain in mind (namely, industry/engineering). This led to the implementation of specialised features such as a map component and the ability to stream properties as they are added (Soylu *et al.*, 2018). These features would not have been implemented if they were developing a visualisation for generic knowledge graphs. By designating this as a domain-specific knowledge graph visualisation tool, we can design interactions that are unique to the domain.

3 Methods

A food waste ontology was created through observation of domain expert annotation of documents. A knowledge graph of individual instances was created from a spreadsheet of human-extracted data and evaluated using visualisation of graph embeddings.

An interactive visualisation tool was developed to support domain expert reasoning using the knowledge graph. Both the knowledge graph and the visualisations were iteratively evaluated and refined in consultation with the collaborating domain experts. The final system was evaluated through user study on independent domain experts.

3.1 Process

This work was conducted in collaboration with domain experts from Wren & Co, a food policy consultancy[2].

Through discursive requirement analysis with the collaborating domain experts, we identified a domain knowledge graph visualisation as the development target. These experts produced a brief seeking an "*informative policy map of UK food policy*" from data they would supply. They wanted a visual representation of the network of directed labelled relationships that exist in food policy. We identified a domain knowledge graph visualisation as a suitable for these purposes. This required development of both a knowledge graph and visualisation tool.

An iterative prototyping methodology (Oates, 2006) with agile approaches was used for ontology, knowledge graph, and visualisation development. Three major review meetings were held with two collaborating domain experts. The focus shifted through the course of the project from ontology development to visualisation refinement. In line with the agile approach, where urgent issues were identified these were prioritised as development targets. The technology deployed was entirely unknown to the domain experts, the domain entirely unknown to the researcher. Hence identification of some development targets was not possible at the outset. As anticipated, the knowledge of both parties evolved through the project. The iterative and agile approach to development meant that this enhanced understanding could be used to refine and refocus the work. This would not have been possible with a more rigid methodology.

Visualisation design was through iterations of Munzner's nested model (Munzner, 2009), iteration frequency, domain-expert involvement and validation approach dependent upon level within the model. This approach leverages the expertise of both parties, without requiring significant knowledge transfer. Domain problem characterisation was performed at the outset, largely by the collaborating domain experts and with limited input from the researcher and is not validated within this work. Frequent algorithm design and validation work was done by the researcher alone, only incorporating feedback from the domain experts when raised. Data/operation abstraction and encoding/interaction technique were also largely researcher-driven, but with expert validation explicitly sought to assess domain validity.

[2] Participant Information Form & Consent Form in Appendix B

3.2 Ontology & Knowledge Graph

Ontology creation was initially performed through observation of expert-annotation of relevant government documents, before being refined in visualisation-led interviews (Hitzler, Krotzsch and Rudolph, 2009). A collaborating domain expert was asked to record themselves annotating three policy documents: Hansard minutes of UK Parliament(*HL Deb. vol. 811 col. 816-819, 24 March 2021*, 2021), A White Paper on Waste (*Our Waste, Our Resources: A Strategy For England*, 2018), and UNEP Food Waste Index 2021(Forbes, Quested and O'Connor, Clementine, 2021). They were asked to identify and categorise things of significance from the text however they saw fit. Viewing these recordings gave us insight into both the final categorisations, and *how* experts reason about this domain. This was used as the basis for a simple initial ontology, which was visualised using CoModIDE(Shimizu and Hammar, 2019), Fig. 2. This was then presented to the domain expert to assess whether it modelled their understanding of the domain. Further expansion of the ontology was needed as more diverse data was added through the project. Proposed additions and amendments that were needed from an ontological modelling perspective were then confirmed with the domain experts, to check for coherence from their domain perspective. The collaborating experts were satisfied that this ontology (Fig. 12) modelled their understanding of the domain.

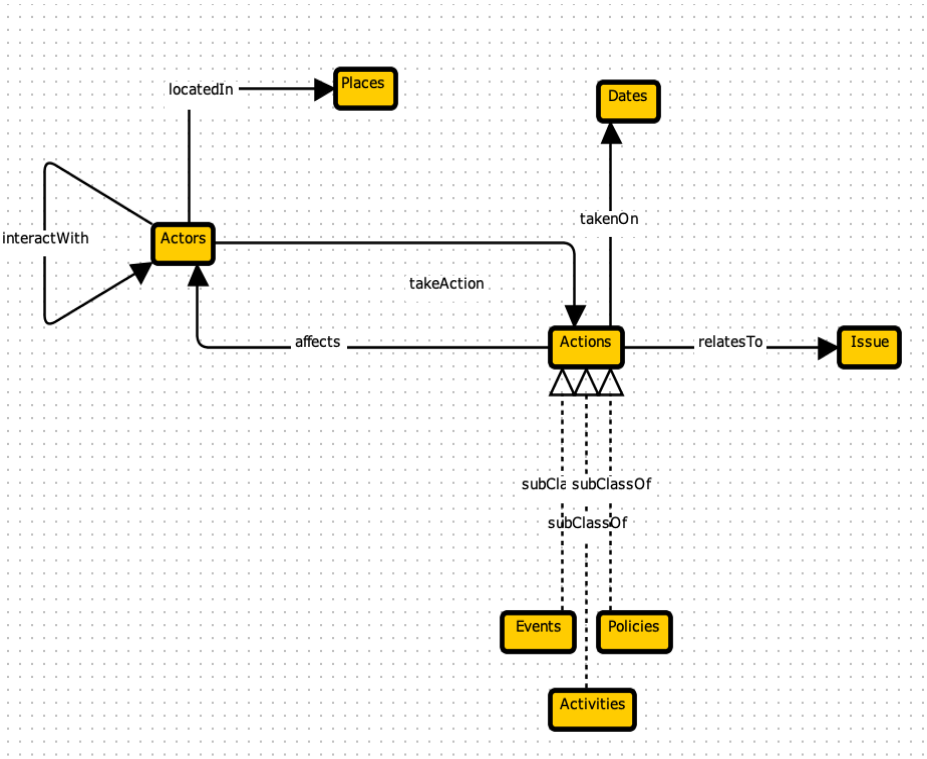


Fig. 2 – Initial schematic ontology

The knowledge graph was created by iterating through tabular data manually extracted from unstructured sources, expanding the ontology in the process. The ontology was used to generate a spreadsheet template, for a domain-specialist researcher to codify unstructured text relating to food waste ("**Dummy Data**", Fig. 3). A second dataset, constructed for a meta-analysis of academic papers regarding food waste (Reynolds *et al.*, 2019), provided a further source of structured data ("**Literature Data**"). For each dataset, each cell within a column corresponded to a named individual, class, object property, or data property. These were extracted iteratively, generating a graph of instances. After extraction, the graph was combined with the ontology. This was then expanded to generate the deductive closure of the set with OWL 2 RL, a simplified version of the Web Ontology Language (OWL), forming the knowledge graph, Fig. 4.

1	Actor 1	Actor 1 type	Creation of t	Action	Action Natur	Location	Date	Impact of th	Actor 2	Actor 2 type	Issue	Value/Weigh	Source
11	DEFRA	Government	launched	Public procur	Toolkit	London	21/04/2014	advising	Catering suppliers	Business	Hospitality a	Support	https://assets.
12	DEFRA	Government	Published	Food Labellir	Guidelines	London	1/11/2017	Advising	Food industry	Business	Food waste (	food is	https://assets.
13	FSA	Government	Published	Food Labellir	Guidelines	London	1/11/2017	Advising	Food industry	Business	Food waste (	food is	https://assets.
14	WRAP	Charity	Published	Food Labellir	Guidelines	London	1/11/2017	Advising	Food industry	Business	Food waste (	food is	https://assets.
15	Champions 1	Coalition	published	Target Meas	Guidelines	The Netherla	1/11/2019	advising	Countries	Nations	Food loss an	strengthen f	https://www.u

Fig. 3 – Sample rows from the "Dummy Data" spreadsheet

```
20203  40 a xsd:decimal,
20204          xsd:integer ;
20205      owl:sameAs 40 .
20206
20207  fp:Advice a owl:Class ;
20208      rdfs:subClassOf fp:Action,
20209          fp:Advice,
20210          owl:Thing ;
20211      owl:equivalentClass fp:Advice ;
20212      owl:sameAs fp:Advice .
20213
20214  fp:Appetite a fp:Journal,
20215          fp:Publication,
20216          owl:NamedIndividual,
20217          owl:Thing ;
20218      fp:impactFactor 2.691e+00 ;
20219      owl:sameAs fp:Appetite .
20220
20221  fp:Catering_suppliers a fp:Actor,
20222          fp:Business,
20223          fp:Group,
20224          fp:business,
20225          owl:NamedIndividual,
20226          owl:Thing ;
20227      fp:activeIn fp:hospitality_and_food_service_food_waste ;
20228      fp:interactWith fp:DEFRA ;
20229      owl:sameAs fp:Catering_suppliers .
```

Fig. 4 – Sample of the final knowledge graph. The displayed triples relate to literal data (40), ontological class (Advice) and entities from the Literature Data (Appetite) and Dummy Data (Catering suppliers). Some of the triples relating to Catering suppliers have been extracted (or inferred) from the data in the first row of Fig. 3.

Ontology verification was performed using automated reasoning to check for logical unsatisfiabilities, and human inspection by both the researcher and domain expert. The expanded knowledge graph was loaded in Protégé (Musen, 2015), to enable the use of a reasoner to check for any statements that cannot be true (i.e. logically unsatisfiable). Pellet(Sirin *et al.*, 2007) was used as the reasoner to accommodate the existence of anonymous individuals in the knowledge graph. Where statements existed that were logically unsatisfiable, either the underlying ontology or the extraction script were amended, as appropriate. Once visualised, both the researcher and domain expert were able to identify further issues using a visual-analytics Human in the Loop approach, which were resolved accordingly. Indicative examples of these are given in Table 1.

Issue	Identification	Resolution
Food Security is an Issue. Food Security is a Publication. Issue & Publication are disjoint classes.	Reasoner	Adjust extraction script to transform all Issues to lowercase.
Michael Gove a Government, Person. DEFRA a Government, Group. Government a Person, Group. Person & Group are disjoint classes.	Reasoner	Change ontology: create 'government' for people in government (e.g. MPs), retain 'Government' for groups (e.g. departments).
Multiple academics named on a single node, split by a semicolon.	Visualisation	Adjust extraction script to include semicolon delimited lists

Table 1 – indicative issues with the knowledge graph, how they were discovered, and how they were rectified.

The created ontology was successfully applied to other documents within food waste, focused on commerce, suggesting generalizability. A collaborating domain expert, who had not extracted data previously, applied the same process to a different data source. The Grocer magazine, a source of commercial information on the same domain was used for this purpose.

Class distinctiveness was assessed through visual analysis of principal component analysis of entities from the knowledge graph vectorized using OWL2Vec*(Chen *et al.*, 2021). Results were broadly positive and are discussed in detail in Section 4.1. This approach allows for assessment of how consistently and distinctively different classes have been modelled within the knowledge graphs. Triples are treated as sentences, and the graph is ingested as a corpus of sentences. Terms can then be vectorized using word2vec(Mikolov *et al.*, 2013, p. 2). If classes of entities are modelled consistently, they have similar vectors. If they are modelled distinctly from other classes, vectors of the two classes are dissimilar. Vectors can be compared pairwise by examining cosine similarities. They may also be compared collectively using visualisation, where principal component analysis provides a 2D representation of all the multidimensional vectors. Here, OWL2Vec* calculated semantic embeddings over 100 epochs, with no axiomatic reasoner or pre-training, a random walk depth of 3, embedding size of 100, and minimum count of 1. Embeddings corresponding to literal data were discarded. Principal component analysis was performed on the remaining URIs to enable visualisation, coloured by superclass.

3.3 Visualisation

General Design

Visualisations were created programmatically using Dash Cytoscape, a python graph visualisation library rendered in browser[3]. The visualisation of a knowledge graph requires the execution of SPARQL queries to extract information from the graph. These SPARQL queries require packages in only python or java. Thus, it was desirable to develop a visualisation in one of these languages and our prior experience with python led to this being selected over java. Dash is a python framework that allows for development of interactive visualisations, rendered in a web browser. Dash allows for dynamic updates to the visualisation from user interaction and integrates html and css to provide further possibilities. The interactive visualisations can then be accessed on any device with a browser and require no specialist skills or setup to access. Dash Cytoscape is a python version of the javascript graph analysis and visualisation library cytoscape.js (Franz *et al.*, 2015). Cytoscape.js offers extensive functionality such as directed edges, hierarchical layouts, edge labelling, and draggable nodes. No other python graph visualisation software, such as pyvis, graph-tool or bokeh, offers all this functionality. Dash Cytoscape was chosen for this work as a python package that allowed for online deployment of an interactive visualisation with the required functionality.

An unbundled, edge-labelled, node-link diagram was used to show the variety of relationship types (Fig. 5). A node-link was selected over a matrix representation because it is more intuitive for a novice user(Ren *et al.*, 2019). The collaborating domain experts had also initially requested a '*map'* of the domain, which can be achieved through a node-link representation. The relationships between nodes are directed, and there are 64 different types of relationship. The diversity of relationships means that encoding them with colour, for example, is not feasible so labelled arrows have been used. Unbundled edges preserve the detail of relationships – their types, directions and number are all visible. While tapered or intensity-varied edges should improve clarity(Holten and van Wijk, 2009), in our experiments the collaborating domain experts found them "*confusing…they don't make it easier to understand, even though logically, I feel like they should*." This makes it more challenging to immediately determine the direction of a relationship. To aid in this, the arrowheads stop short of the target, while they start *at* the source. Further, where arrows are curved this is asymmetric.

[3] Visualisation URL and Code folder details given in Appendix F

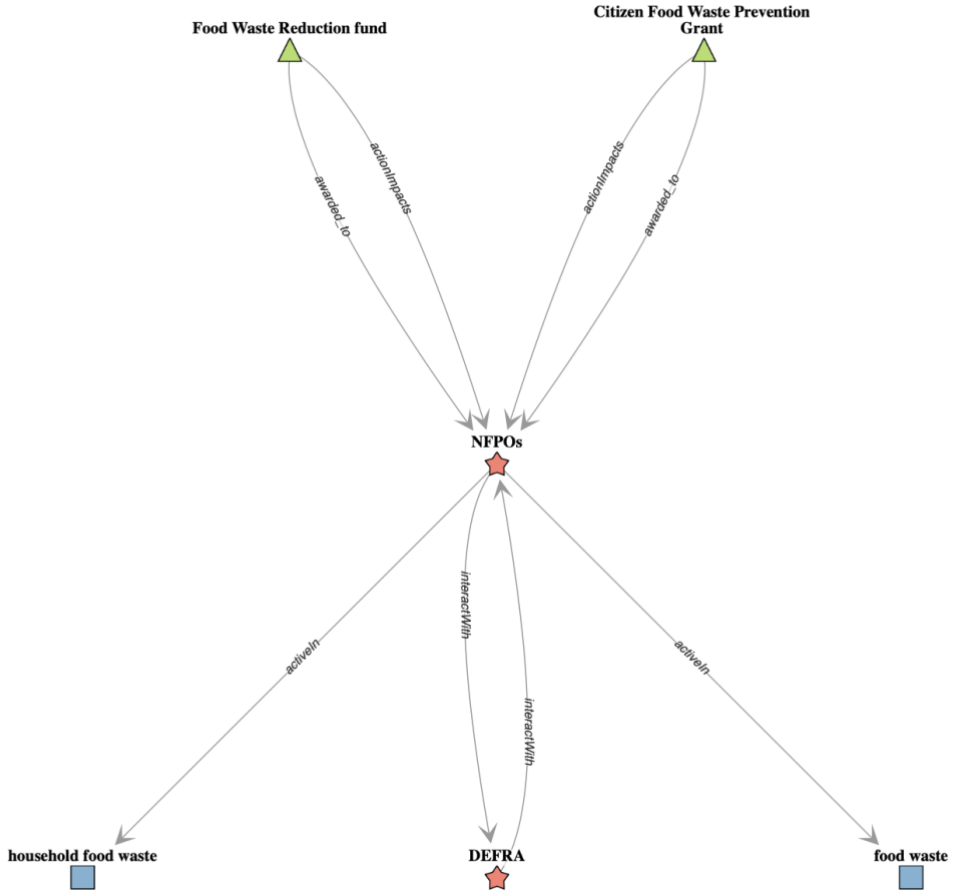


Fig. 5 – The node-link representation used in the final visualisation. This small network shows everything that is directly connected to the node 'NFPOs'

A categorical colour scheme was used to distinguish between node superclasses, with node shapes used as a secondary encoding to improve accessibility. Class membership is very important, but it was not practical or sensible to represent all 88 classes on the main visualisation. The six superclasses, plus literals (descriptive information), provide a good level of distinction for encoding. A categorical colourscheme was selected using colorbrewer (Harrower and Brewer, 2003) to represent the disjointedness of classes. The number of classes meant that a colour-blind accessible scheme was not possible, and so node-shape is a secondary encoding to aid. Some nodes were difficult to see on the white background, so a black edge-line was added. One or more nodes can be selected and are then highlighted, which helps keep track of important nodes upon layout changes. The most recently selected node's incoming and outgoing edges are also highlighted in different colours (from a similar part of the colour palate) to help with understanding in larger views(Ware and Bobrow, 2005). A legend is available in one of the tabs at the bottom (and displayed by default on loading the visualisation), with both colour and shape represented, Fig. 6.

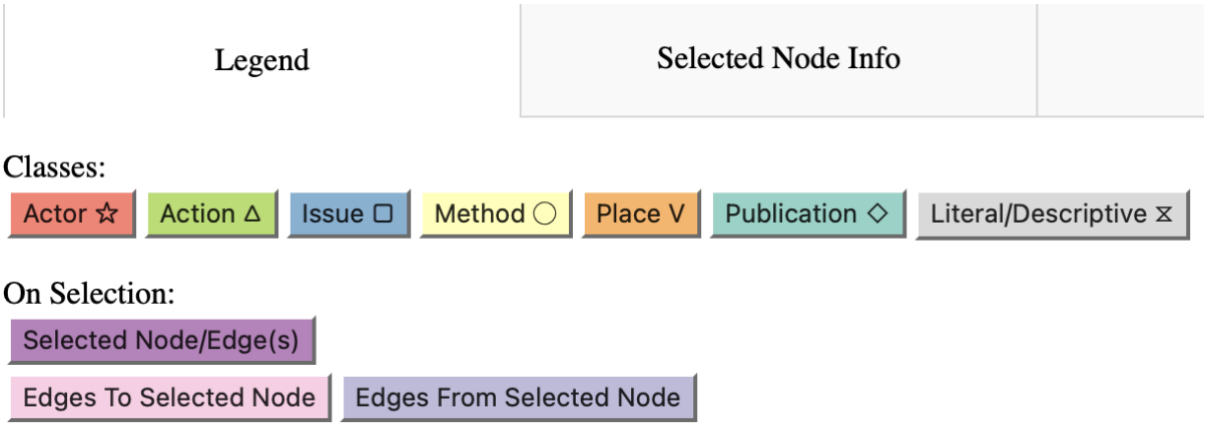


Fig. 6 –The Legend. In addition to colour, symbols are given to represent the shapes of nodes.

Full node details are available on demand via the 'Selected Node Details' tab, Fig. 7. The full name (or text, where literal) and all class membership (including all subclasses) of the last node selected are visible in the 'Node Detail' tab, which are not always shown elsewhere.

Node and edge labels are only displayed when above a threshold size, to improve clarity for larger views. Without this, large network views are cluttered with text which occludes both nodes and edges. The labels are only displayed when above the first point size where they become adequately legible. This means that little is lost by hiding them below this size, as the viewer would be unable to read them anyway.

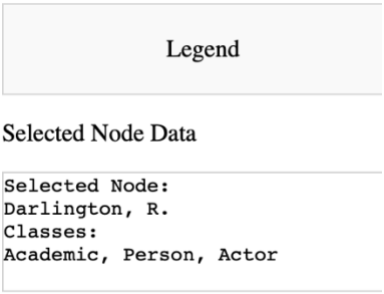


Fig. 7 – Details for a node

Multiple differences in label font make it easy to distinguish between node and edge labels, Fig. 8. A larger, bold, serif font is used for node labels, while a lighter italicised font was used for edge labels to emphasise the transitionary nature of the relationships they represent. Edge labels are aligned with the direction of the edge, while node labels are always upright.

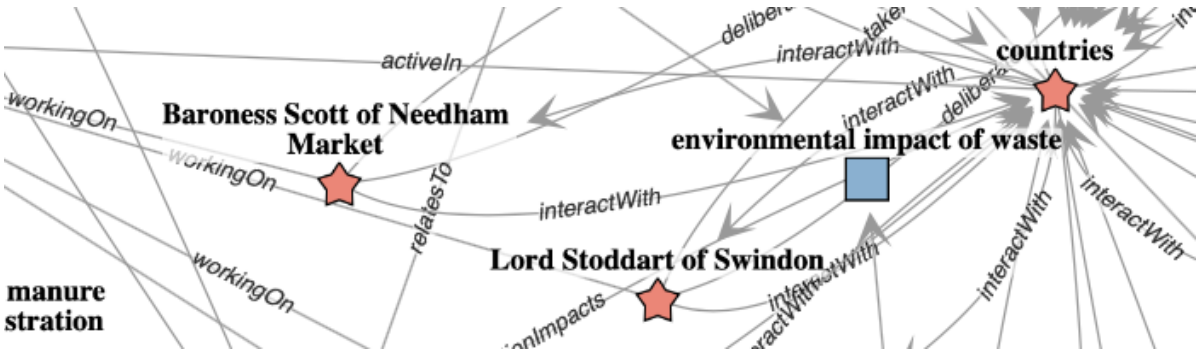


Fig. 8 – A cluttered section of graph. Even when there are many labels on display, the node and edge labels are easy to distinguish from each other.

A modifiable CoSE Bilkent layout(Dogrusoz *et al.*, 2009) was used for views with over 200 nodes, and a modifiable Dagre layout for those with fewer. The CoSE layout results in close proximities for related nodes. For overviews, this helps find areas of interest and cliques. The domain experts wanted to understand why some nodes ended up close together. For smaller networks, the hierarchical Dagre layout highlights directionality of relationships. This helped users trace impacts through small networks. Users can move nodes to adjust layouts, as this process of interaction has been shown to improve understanding and user satisfaction(Dwyer *et al.*, 2009).

Users can filter by node and edge types, with some hidden by default. The "Display Options" tab allows users to select which classes and edge labels to show or hide, Fig. 9. While Schneiderman's mantra(Shneiderman, 1996) suggests a full initial overview Fig. 10a, the domain experts found this overwhelming and incomprehensible. They requested that this only display the Dummy Data, and the Issues from the Literature Data, illustrating their disconnectedness from policymaking, Fig 10b. This is achieved via a to show all data, or exclude the classes exclusive to the literature review, with the latter selected by default. Nodes and edges indicating class membership are not typically shown, as it would cause other nodes to cluster around their respective class nodes and be detrimental to interpretability. These *are* shown when focussing on nodes, as this is intended to provide greater depth.

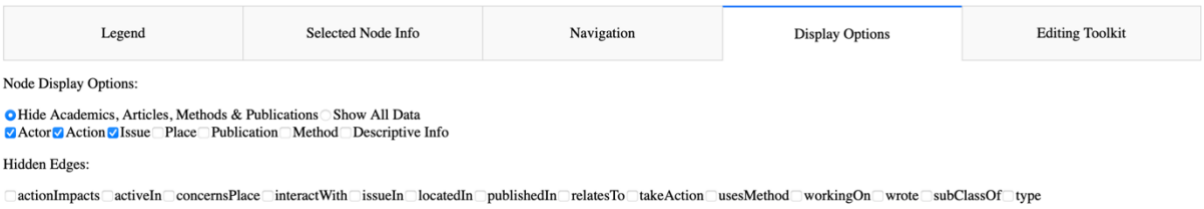


Fig. 9 Display options to adjust which nodes and edges are shown in the main visualisation

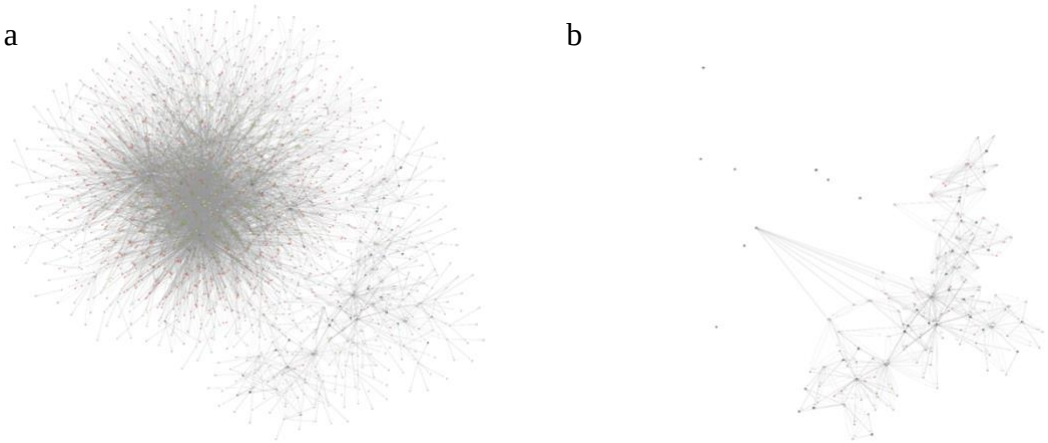


Fig. 10a Overview of all data, and Fig. 10b with Literature Data hidden.

Explore

Three overviews are available as a starting point for exploration via the 'Navigation' tab (Fig. 11): entire dataset (Fig. 10a&b), class hierarchy (Fig. 12), and class schema (Fig. 13). The 'entire dataset' view displays all nodes and edges that have not been filtered using the display, excluding class-membership relationships. The 'class hierarchy' view is a visualisation of the class structure in the ontology. The 'class schema' view adds edges with indicative relationships between classes. These are not inherently present in the extracted data and were manually added. For the latter pair of views, positions are explicitly encoded, to emphasise the class structure.

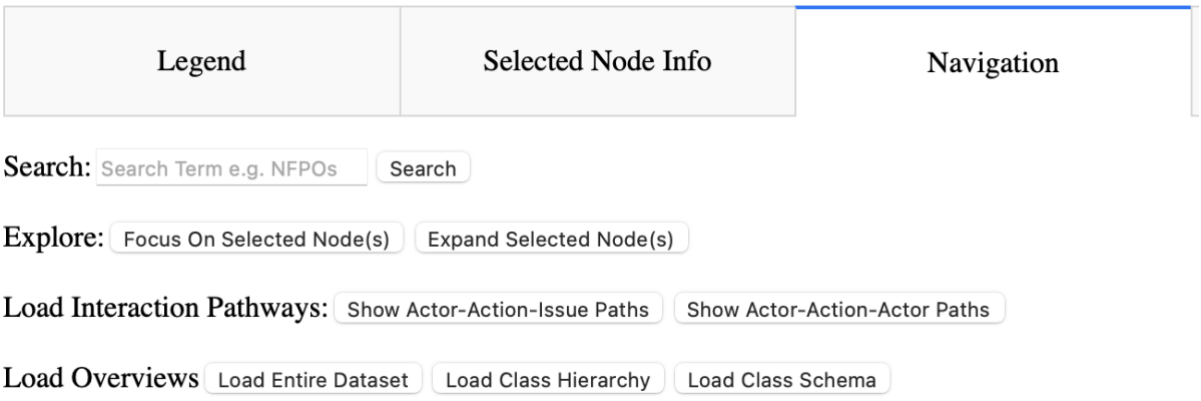


Fig. 11 – The Navigation Tab

Users can explore through focusing on or expanding a selected node. Focusing visualises that node and its direct connections. Expanding retains the current view and adds the selected node's direct connections. In both cases, the display rescales, and layout recomputes. The selected node remains highlighted to reduce the dissociative effect from this change of layout.

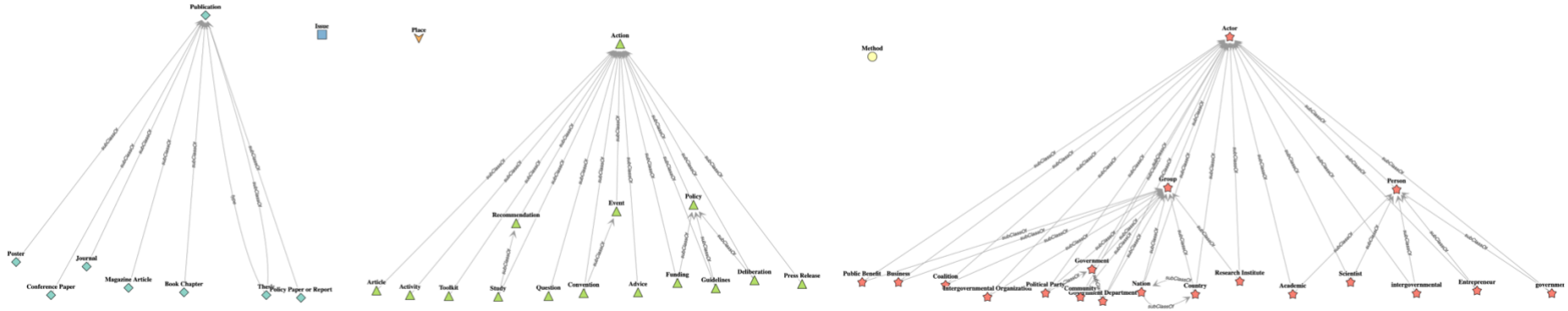


Fig. 12 – Class Hierarchy view, visualising the structure for the ontology

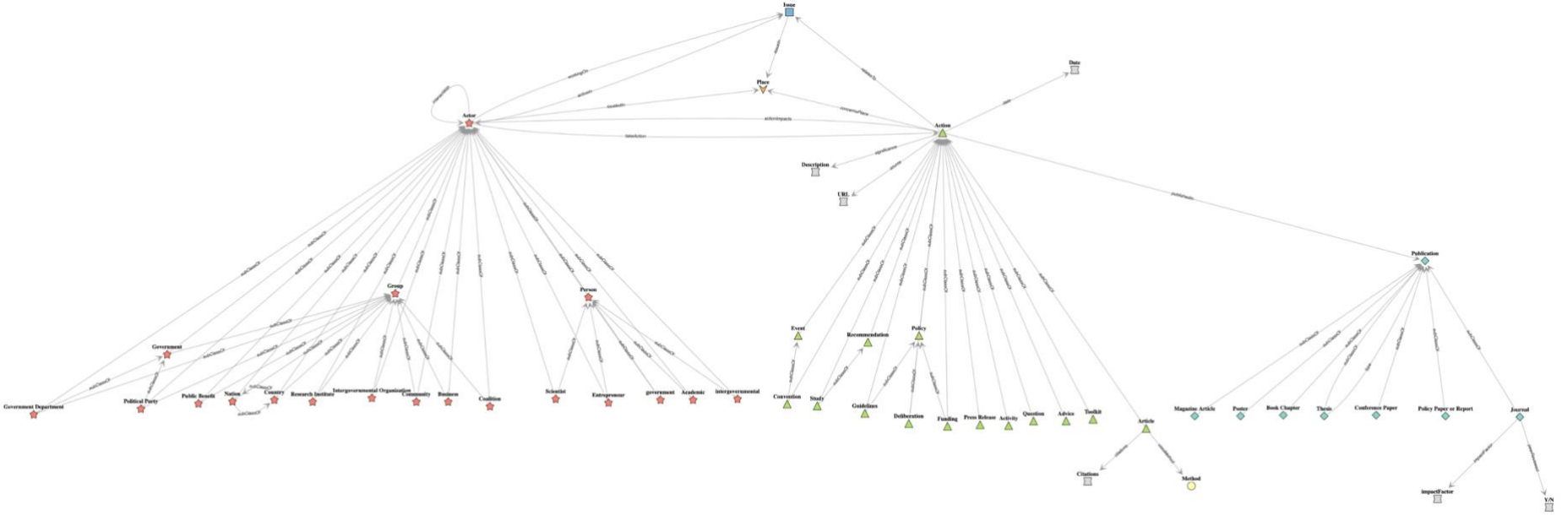


Fig. 13 – Class Schema view, as per the Class Hierarchy view but with edges and nodes manually added to indicate typical interactions and literal data

Search

Users can search for nodes, with the closest lexical match being returned. Searching enables users to quickly focus on their area of interest. Feedback on early builds which required an exact match indicated frustration with knowing '*what to search for'*. This was compounded by spelling errors and inconsistent capitalisation in the underlying data. The deployed version compares lexical similarity to find the most similar node & its connections. This does not consider node class or meaning so, while it will always return a result, it may not be relevant. For example, a search for '*banana*' will return everything connected to '*Canada'*.

Create

Users can add nodes with classes and directed, labelled edges. In line with the open-world assumption(Allemang and Hendler, 2011), the knowledge graph is considered incomplete. Many visualisations present the information in the underlying knowledge graph as in a manner that loses this idea of incompleteness. Here, the knowledge graph visualisation is also presented as incomplete and any user can add anything to it. To add a node, users enter its name and list of classes in the editing toolkit (Fig. 14). This creates a disconnected node, styled according to the listed classes. Edges are added by selecting two nodes and naming their relationship. In both cases, these are added to the visualisation, but the underlying knowledge graph is not changed. The decision to keep 'edit' functionality at the visualisation level was to avoid problems that could arise from non-technical users editing the knowledge graph. For example, a user could create a node with the classes Person, Government – a logically unsatisfiable statement (Table 2) that would break the knowledge graph. This decision also allows users to create 'views' of interest to their specific use through adding or deleting information from the visualisation without affecting the integrity of the data for other users.

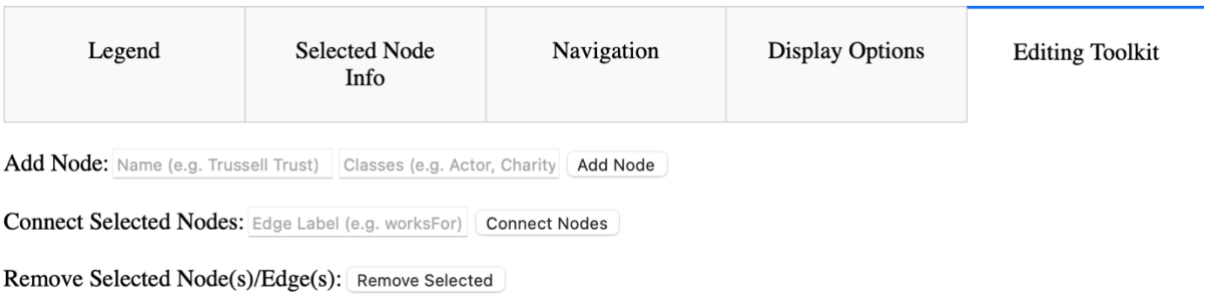


Fig. 14 – The editing toolkit provides add node, add edge, and delete functionality.

Manage

Users can remove selected nodes and edges, allowing indirect editing of all visualisation elements. There are errors in all but the smallest knowledge graphs(Paulheim, 2016), and so users can delete incorrect or irrelevant information from view. This, combined with the 'add' operation provides an interface for users to edit the visualisation. This provides 'canvas' functionality for users to create custom visualisations that can then be saved and used elsewhere. Again, the underlying knowledge graph is not changed.

Domain Specific Tasks

Domain-specific views are available from the navigation tab, visualising property-paths considered important by the domain experts. Many pre-existing tools offer support for users to build their own SPARQL queries, as outlined in Section 2.2. Conversely, we predefine queries associated with two domain-specific interaction pathways highlighted as important by domain experts, Table 2. There are buttons in the 'Navigation' tab to visualise all of these 'Actor-Action-Actor' and 'Actor-Action-Issue' paths. For Actor-Action-Actor, Fig. 15, a Dagre layout emphasises the sequential nature of these pathways. For Actor-Action-Issue, Fig. 16, CoSE Bilkent shows cliques and key issues. This was found to be of particular interest to domain experts, who wanted to understand reasons for differences in connectedness.

Actor createsAction Action .	Action relatesTo Issue .
Actor createsAction Action .	Action actionImpacts Actor .

Table 2: The pairs of triples to generate property path visualisations

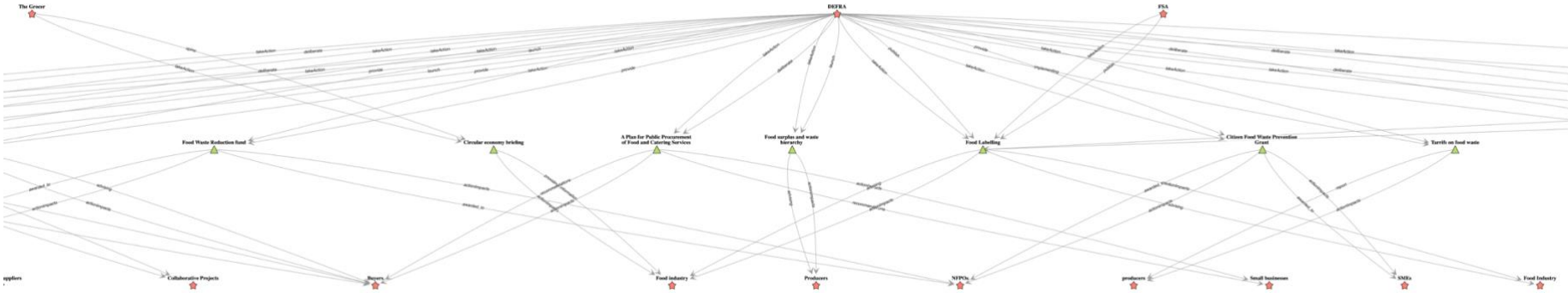


Fig 15. A partial zoom view of the Actor-Action-Actor pathways

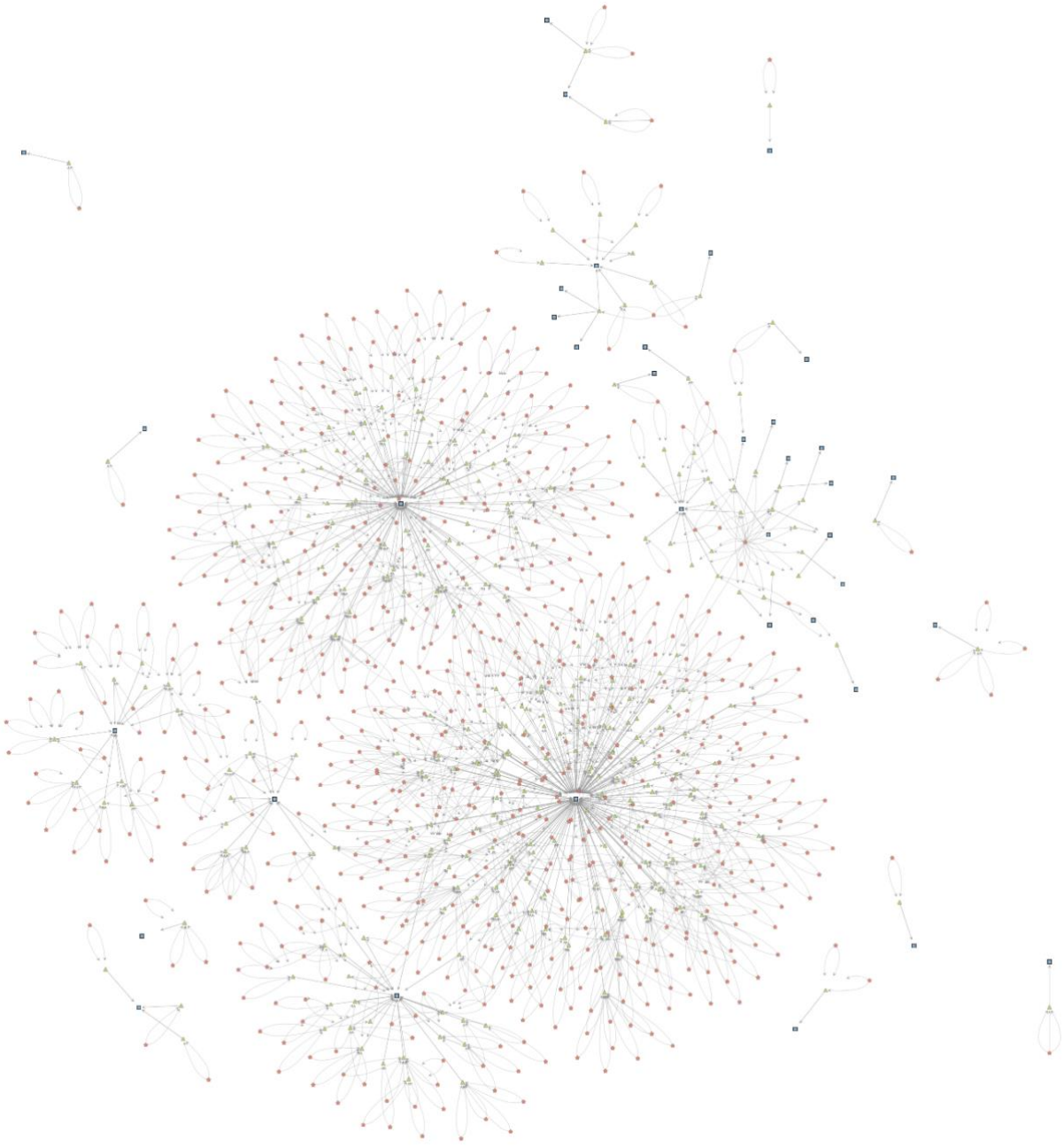


Fig 16. The Actor-Action-Issue pathways.

Algorithmic Efficiency

Performance has been improved by storing a second dataset of all nodes with precomputed positions in memory. This improves load times by 2 orders of magnitude. Extracting all nodes, classes, and relationships directly from the knowledge graph and calculating layout took >5min. Using a separate dataset of precomputed positions reduces this to ~3sec. Once loaded, filtering and other minor interactions are performed on the cached dataset and so also much faster. Other interactive operations involving loading a new subset of the data, such as search or expand, are executed via new SPARQL queries with subsequent recomputation of layout. Due to the comparatively small size of these subsets in all but one case, this approach is not detrimental to performance. The exception, with a wall clock time of ~30 seconds, is the Actor-Action-Issue view, due to the number of results for this query.

Further performance gains were achieved through deploying vectorized approaches. Serializing SPARQL output must be done iteratively. Retrieving classes is achieved through executing SPARQL queries on each of the returned terms. String methods must be performed on all these outputs, making output very slow if done during serialization, using a vectorised approach notably improves performance.

A further dataset of all nodes is stored to enable lexical similarity search. Levenshtein distance from the entered search term is calculated for all nodes and normalized using the longer term length. The lowest-scoring term is used in an exact-match SPARQL search of knowledge graph.

Evaluation

The visualisation utility is assessed against the 15 use cases defined by Po et. al (Po *et al.*, 2020). Across the 15 use cases, 29 knowledge graph visualisation tools were evaluated. These use cases are constructed to represent the typical tasks that a linked data user may seek to accomplish. These include "*The visualisation of the instances belonging to a specific class*" and "*the visualisation of statistics about the dataset*". Each task was abstracted into an activity diagram representing how a user might accomplish that task. Each of the tools was then assessed as to whether each task could be accomplished. The tools have a median task completion rate of 3, with a maximum of 8 and minimum of 1. Performance relative to these is detailed in Section 4.2.

Final validation was performed as user-study involving five independent domain experts[4] unfamiliar with the tool to assess usefulness and usability. Five 30-minute virtual interviews[5] were conducted with food policy experts, two specialising in food waste. None had any experience with knowledge graphs. Participants were asked to share their screens and access the software. Sessions consisted of:

- a short introduction
- guided-user tasks
- free use of the tool
- question-led discursive feedback

Participants were asked to stop screen sharing and complete a System Usability Scale (Jordan *et al.*, 1996) questionnaire online. The sessions were recorded using audio-only to protect participant privacy.

[4] Participant Information Forms & Consent Forms in Appendix C
[5] Full interview Protocol in Appendix D

Guided tasks provided participants with an introductory tutorial of the system while measuring tool effectiveness and efficiency, with further questions prompting human reasoning and reflection facilitated by prior domain knowledge. Users were provided with tasks to complete, under the guidance of the researcher, covering the full gamut of tasks typically required of semantic web visualisations (Pesquita *et al.*, 2018): Explore, Search, Create, Manage. These tasks were designed to provide a 'tutorial' of the functionality of the software, with task completion recorded to measure tool effectiveness and efficiency. The tasks were supplemented by questions designed to require human reflection and reasoning that would not be possible without both visualisation and prior domain knowledge.

Example Task: "*Identify one piece of information/statement now available from this view*"

Example Reasoning Prompt: "*Please comment on that statement.*"

Discursive feedback, after the participant had been allowed to use the tool as desired with support from the researcher, provided qualitative feedback on the utility of the software. Free-use of the software (with support) was important to allow the participant to establish how it may be of interest to them. Users were then asked 5 questions to gather qualitative feedback on the software:

- Has this tool confirmed anything you already knew?
- Have you learned anything new or useful from using this tool?
- Has this tool affected how you think about food policy? How so/how did it achieve this?
- What aspects of the tool (graphics/interactions) do you think would be most useful to you? Why?
- Would you use this tool regularly to support your work? How?

Transcription and full, formal thematic analysis were deemed beyond the scope of this work, due to the short time frame of the project. Nevertheless, the audio recordings of the sessions were used to identify common themes and extract indicative anonymised quotes from participants.[6]

[6] Interview notes submitted in Additional Files

4 Results

The ontology was instrumental in the reflection and reasoning process of the experts and generalizes beyond food waste. We found that the knowledge graph captures the nature of Actions and Actors appropriately, but other classes are less-well defined.

The use of a node-link diagram and design choices supported participants in using the visualisation. The visualisation offers greater functionality than any other knowledge graph visualisation tool evaluated by Po *et al.* (2020). *Independent domain experts were most interested in explore, search and domain-specific interactions. These experts were impressed with the utility of the system, were able to use it to support their reasoning, and could see themselves using it in their work.*

4.1 Ontology & Knowledge Graph

The ontology, although not explicitly identified, was instrumental in the reflection and reasoning process of experts and generalizes beyond food waste. All 5 experts highlighted the classes Actor, Action & Issue as one of the more interesting takeaways from using the tool. They were not asked a question that explicitly targeted this ontological modelling, but raised it independently. For one non-food waste expert, this categorisation was entirely new and revelatory: "*I think actor action issue is a nice way of explaining and showing kind of the complexity of the different components. So, that for me is a new way of looking at this*". Another who was more familiar with causal loop diagrams felt "*These first three: issues actor action, […]is possibly the most useful area. The mapping of those three classes is particularly helpful*." It is clear that this ontology has successfully modelled the domain understanding of food policy experts, beyond the food waste data it was designed with.

The final knowledge graph is relatively small and simple, and so has only captured some class distinctiveness. The final KG consists of just 20556 facts about 2610 individuals, across 6 superclasses and 82 subclasses. The subclasses are unevenly distributed – Actor has many subclasses, Issue has none. PCA of graph embeddings, Fig. 17, shows that Actor and Action are very distinct from the other classes who overlap considerably. This suggests that these two classes are modelled differently from the other classes in the graph. The greater isolation of Action indicates that this class is particularly unique. Action also consists of a single, fairly small cluster, indicating consistency in how the class has been modelled. Actor embeddings are more spread out, indicating differences in their knowledge graph environments. The class consists of three clusters and a more diffuse region. Inspection of the three dense clusters of Actor do not indicate any apparent differences in membership – they appear to all be academics, originating from the Literature Data. The more diffuse cluster of actors around (0,1) represents the other Actors, e.g., politicians and organisations, from the Dummy Data. There is considerable overlap between the remaining classes, suggesting that they are not well-defined in the knowledge graph. This may be due to the shallow class structure for these classes, or lack of triples concerning them compared to Actions & Actors. In both cases, this is likely due to the limited data used in construction of the knowledge graph.

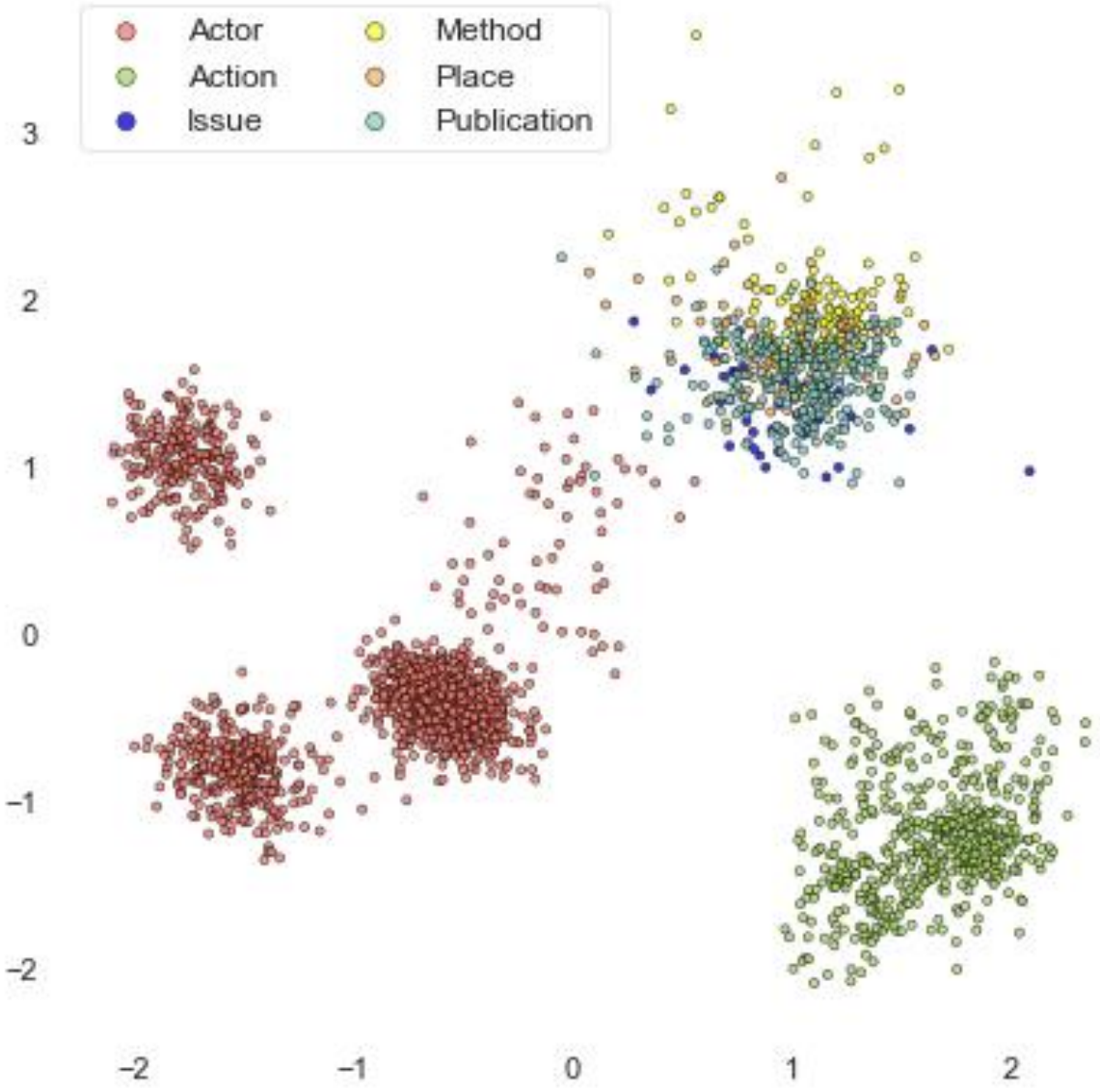


Fig. 17 – PCA of FWKG Embeddings

4.2 Visualisation

The use of a node-link diagram and design choices supported participants in using the visualisation. All participants were able to extract information from the visualisation. Task one completion by all with minimal prompting demonstrated that the use of node colours and shapes for superclasses was appropriate, and that the legend was easy to understand. Participants were also all easily able to complete task 3 and identify the statements represented by a connected pair of nodes, suggesting that the node-link representation was interpretable, and directed labels were sufficiently differentiable. The short-stopping of arrows was useful: *'I guess it's pretty consistent that the lines – if it's coming from something it originates from [it's] touching and then the arrow doesn't go to [the target] so I think that was pretty clear*". One participant was initially confused over which node an arrow terminated at, but this was rectified once they clicked and dragged the node to move it.

Details of node subclasses were understood by all participants, but the utility of this is questionable. All participants could use the 'Selected Node Info' tab to acquire this information, but it was not used in their reasoning process. Only one participant returned to this tab in subsequent tasks after being shown it.

Understanding aspects of the class hierarchy was challenging for all participants. The task participants consistently found most challenging involved using the Class Schema view to understand the difference in categorisation of the two subclasses 'government' and 'Government'. This task is challenging, representing an understanding of an ontology rather than relatively simple relationships. It is more abstract than other views using the visualisation. Nevertheless, all participants could understand and reason using it. One waste expert requested further time to understand it more fully after the session.

Visualisation display options were not used by participants except for when directed to do so. The options to hide types of edges were never used. The node display options were not used by participants after being told to turn them all on. Some wanted to things that were supported by these display options, such as displaying all issues, but did not think to use them to accomplish this. They were able to accomplish these goals under the direction of the researcher.

The layout choices helped participants in making inferences using the visualisation. Most views were in the Dagre layout, resulting in a downward flow of interactions. This was not explicitly conveyed to participants or discussed, but they had learned this and were using it in the later tasks when appropriate. This was especially apparent in the 'Actor-Action-Actor' task, where participants found it very easy to identify which Actor was the source of Actions and which Actor was impacted by them. "*Looks like higher up is giving and lower is receiving*". *"In terms of generating the actions DEFRA is the main player."* The CoSE-Bilkent layout was also appropriate for the bigger views, again exemplified by the Actor-Action-Issue view, where participants could immediately identify highly connected issues and those that were much less connected. Perhaps consequentially, reasoning was rich and well beyond what was represented "*Household food waste is a large demographic and possibly more measurable […] whereas dumpster diving seems like it would be a small demographic, small use case, but also hard to measure."*

Four of the five participants completed all tasks across the spectrum of Explore, Search, Create, Manage, and domain-specific tasks. This suggests that the visualisation is broadly effective at supporting the typical user tasks required of semantic web visualisations.

Create and Manage functionality seemed to be of little interest to the domain experts. All could complete these tasks and thought went into identifying relationships to add and data to delete. However, there was very little discussion of these at the time and was not raised by any participants as a valuable feature during discursive feedback. It seemed that participants were more interested in using the tool to extract accurate information than correct or add to it.

The two domain specific interaction pathways seemed to be the most interesting views. Participants were also able to suggest reasons for these using their own knowledge, sometimes with deep thought processes. "*Household food waste is very well connected – there's an, awful lot of it so lots of action focussed on it. I'm surprised there isn't more on anaerobic digestion - potentially because it's less impactful. A lot has been going into it, but it's a less tricky issue […] businesses have just been getting on with it. Household is multifaceted […] so there's lots going in from lots of different actors. Wow! It really is quite complicated.*" suggesting this was supporting their reflection and reasoning purposes.

One participant could not complete all tasks due to processing issues, suggesting further algorithmic optimisation work is needed. They were running other processes and had other tabs open in their browser, resulting in insufficient memory for some tasks. Some tasks are completed client-side, most notably layout computation, which can be very intensive. This layout computation is unavoidable as data is retrieved dynamically from the knowledge graph. While all other users (including early tests on the collaborating domain experts) had not raised this issue, this suggests that further improvement to algorithmic efficiency is needed.

When allowed to use the tool freely the waste experts sought to explore overviews, while non-waste experts searched for specific nodes. The waste experts, independently, wanted to see all instances of a class and expressed a desire for a tabular display, with one also wanting a 'league table' for connectedness. Other experts searched for things, seemingly hoping to use the tool on their sub-domain.

Non-waste experts were more confident in their reasoning than the waste-experts. The waste specialists were more evaluative of their discoveries, commenting on validity. Some reinforced their prior understanding or '*made sense'*, while other results were '*surprising*'. Other experts were more inquisitive reasoning in the new sub-domain, happily 'assuming' or 'guessing' using the tool.

The visualisation can satisfy 9 of the 15 use cases, more than any tool evaluated by Po *et al.* (2020), and also satisfies further domain-use cases. Use cases 1, 2, 4-6, 8, & 11-13 are achieved[7]. The domain-specific Actor-Action-Actor and Actor-Action-Issue views may be considered as inverted application of Use Case 7: rather than extracting the path between two instances, these extract all instances connected by a given path.

The mean system usability score was 75, in the 73rd percentile of usable systems. While this sample is small, this score of above 68 indicates that the system is seen as adequately usable by participants.

[7] Appendix H

All users found the visualisation useful, could see themselves using it in their work, and the interactions supported information seeking. Participants found directed relationships and visualisation of the domain's complexity very useful. Waste experts, who were familiar with causal-link diagrams, highlighted the tool's potential for discovering unknown entities of significance, directing subsequent work, and "changing the way people think about [food policy systems]". Non-waste experts thought it would be useful as a presentational device, and for establishing strategies. Compared to traditional dashboards this visualisation "is much clearer for establishing relationships between the different players […] there's a huge advantage to it."

Users suggested improvements to the software, with the only consistent theme being different underlying data. All participants suggested that the tool could be of use to them or their organisation if the data it represented were different. For food waste experts, this was data in greater depth. For non-waste experts it was data relating to their specialism. Other suggestions were raised by individual participants but were not corroborated by others. One participant asked for larger icons and a more vibrant colourscheme. A second asked for a permanent display of the legend, and for outward links to reference documents. Another was confused by the class membership edges and nodes that become visible when focussing on a node, suggesting that their complete removal from all views would be beneficial. Despite being anticipated to be problematic, no users raised issues with the layout change upon node expansion.

5 Discussion

The simple ontology was appropriate for this work and generalises beyond food waste. The small domain knowledge graph could be of great use if enhanced with further data.

Focussing knowledge graph visualisation development on specific users facilitated its success, suggesting this approach should be used in other work.

Parallel development of a knowledge graph and its visualisation, with the latter prioritised, improved development efficiency and visualisation effectiveness. This novel approach merits further research.

Developing domain knowledge graph visualisations provides opportunities to advance work in both fields.

5.1 Domain Knowledge Graph

The simple ontology developed was appropriate for the scope of this work is applicable beyond food waste. The classes Actor, Action and Issue were recognised by all the independent experts as a highly valuable way of describing the domain. The ontology was recognised as applicable to areas of Food Policy outside waste. The core interaction pathway:
Actor createsAction Action .
Action impacts Actor .
Action relatesTo Issue .
appears generalisable outside of food policy. The ontology could potentially generalise to Policy in general, or wider Food Systems. This could be established through discussion with experts and applying it to data from these domains.

The structure of subclasses in the ontology may be more complex than required. While participants found the superclasses useful, the only sub-class that appeared so was Academic (a subclass of Actor). In all other cases, while participants could identify multiple classes using the Selected Node Info, they were not useful. This was not normally new to them, and it was never of interest. This may be because of their expertise, and so may be useful for novices. However, it may be worth simplifying the ontology to remove many subclasses. This would make categorising automatically extracted entities simpler and reduce processing complexity.

With further data, this knowledge graph can be a valuable resource for practitioners in the domain. As established in Section 2.1, there is an unmet need for a knowledge graph of food policy. Domain experts felt the knowledge graph modelled their knowledge and understanding but was not yet useful. Those who specialised in Food Waste found the data too simple, and lacking depth for themselves. They felt it would be "*really valuable*" for someone coming into the area and used "*almost daily"* by someone in their organisation. Other domain specialists had little interest in food waste but felt with the right data "*it would be very, very useful*". Both groups could imagine the tool being useful with other data, or for other groups. The project to automatically extract more relevant data is ongoing, with a view to ingesting this into the knowledge graph. Beyond improving its utility, this should improve class distinctiveness and so its quality. Given the domain expert feedback, and their diversity of expertise, a more developed version of this domain knowledge graph is likely to be widely used.

5.2 Knowledge Graph Visualisation

Developing a specific visualisation resulted in better outcomes for users than would have been possible with a pre-existing tool designed for general use. As mentioned in Section 3.1, Wren & Co wanted "*informative policy map of UK food policy*" necessitating an entity-level node-link diagram. Of the visualisations in Section 2.2, only four offer this approach. LODmilla (Micsik, Turbucz and Tóth, 2015) cannot be used beyond its two predefined datasets. RelFinder (Lohmann *et al.*, 2010), LodLive (Camarda, Mazzini and Antonuccio, 2012) and Tarsier (Viola *et al.*, 2018) do use this approach, but fail to satisfy the user needs for other reasons. Tarsier is very difficult for novice use, and neither RelFinder nor LodLive offer the ability to view ***all*** instances at once. This functionality is rare in visualisation tools. It was defined as Use Case 4 by Po et. al. (Po *et al.*, 2020), with only two tools able to accomplish it. The first, VisGraph^3 (Tomaszuk and Truchan, 2017) *only* meets this use case and no others, making it of very limited use. The second, graphVizdb (Bikakis *et al.*, 2016) does not offer explore functionality (such as search, or focus) making it of little use for extracting specific, detailed information. No tool exists that can satisfy the needs of all users, all the time. This visualisation was designed for specific domain-expert end users, for the dataset provided by Wren & Co. It was broadly effective for their needs but would not meet the needs of a different group of users. It is prudent to design knowledge graph visualisations for a specific group, rather than trying to satisfy all possible users.

There was not sufficient time in the interviews for users to fully evaluate the software. The sessions only allowed for a brief introduction to the tool, so users did not have time to use all functionality. Support was given to users through the initial tasks to help them to complete them within the timeframe of the interview. This ensured that users had a full 'tour' of the software but precluded the domain experts working out how to use the software for themselves. This could have been useful for deeper evaluative purposes, such as measuring task completion times. Free use and qualitative feedback were also limited by time. During free use of the software, it was only possible for users to complete one 'task' that they were interested in. Several users wanted to do more. Participants therefore had very limited exposure to the tool to feed back on. The time available for meant that follow up questions to participant feedback could not be asked, further limiting the depth of feedback. Further user study with this tool is valuable future work, with significantly more time allocated to each participant.

5.3 Prioritising Visualisation

Development of the knowledge graph in parallel with its visualisation significantly altered their design. A plausible approach to the project could have been to spend significant time developing a knowledge graph to satisfy as many of the 69 linked data quality metrics (Zaveri *et al.*, 2015), and then develop a visualisation to adequately represent it in the remaining time. This is likely to have resulted in knowledge graph development choices that were detrimental to the goals of visualisation (such as shortening URIs which would compromise interpretability). It *would* have reduced the time that was available for visualisation development, and so limited what could be achieved within the project. Indeed, most of the visualisations developed a posteriori can accomplish far fewer use cases than ours(Po *et al.*, 2020), despite larger teams and longer development. The novel approach taken prioritised the visualisation and built a knowledge graph to facilitate this goal. This resulted in more efficient development, and a more effective visualisation.

Some of the use-cases were only accomplished because of the design of the underlying knowledge graph. One such example is Use Case 2, Visualising the Information of a Class. In a normal knowledge graph this could include things like descriptive text. These are not present in this knowledge graph by design, as there was no intention to visualise them. If such information were to be added, it would be possible to modify the visualisation to display it. This demonstrates the benefit of our novel approach of designing a knowledge graph to facilitate the visualisation. Construction of knowledge graphs in parallel with their visualisation, prioritising the latter, offers a new approach to development and merits further research.

5.4 Visualising Small Domain Knowledge Graphs

The rich features implemented in the visualisation were possible due to the small knowledge graph and may not translate to larger ones. More use cases could be accomplished with this visualisation tool than any analysed by Po et al. (Po *et al.*, 2020). This was partly due to the size of the knowledge graph. This knowledge graph consists of approximately 2500 individuals, whereas most are several orders of magnitude larger. The small graph has allowed for performance optimisations through storing multiple datasets, including precomputed positions for all nodes on overview. This would not be possible for larger graphs. Even on this small dataset, performing SPARQL queries and calculating layout for Actor-Action-Actor/Actor-Action-Issue pathways can take up to a minute. Significant optimisation work would be needed to use this on larger graphs. Some visualisation features may not scale up well. Some of the more connected nodes, such as DEFRA, are already difficult to make sense of. Similarly, views such as the Actor-Action-Actor pathways may stop being usable in their current form. While work may be necessary to accommodate a larger dataset, the fundamental design principles are likely to translate.

Domain-specific interactions were invaluable, and the development of further domain knowledge graph visualisations merits further work. Domain-specific knowledge graphs have only very recently been defined (Abu-Salih, 2021). The field of domain-specific knowledge graph visualisation is close to non-existent, with only OptiqueVQS (Soylu *et al.*, 2018) really having any domain-specific functionality. The domain expert users appeared to find the most value from the two views that were tailor made for this domain. These were a relatively late addition in development, only made possible through the close collaboration of domain experts with the researcher. These were predefined, based on domain understanding. Their execution requires no technical knowledge such as structuring a SPARQL query. There are likely to be further interaction pathways within food policy that can be elucidated through further work with domain experts. Other domains will also feature their own interaction pathways. Finding these requires ongoing collaboration between developers and domain experts, but they have the potential to enhance understanding in both domains. Collaboration between domain specialists and visualisation practitioners offers a rich seam for research.

6 Evaluation, Reflections & Conclusions

This work has been largely successful in achieving its research goals, despite significant difficulties. The work is of use to both the food policy and visualisation domains, making novel contributions to both. This work was also of great personal benefit to the researcher.

6.1 Research Question & Objectives

The project aimed to answer the following research question:

RQ: How can a domain knowledge graph of food policy be visualised effectively to support the reflective and reasoning process of domain experts?

The project outcomes were achieved in pursuit of the following research objectives:

RO1: To develop an ontology in collaboration with food policy experts and using this to construct and refine a knowledge graph of the domain.

RO2: To design, implement and evaluate interactive knowledge graph visualisations that effectively support the reflective and reasoning process of domain expert users, and can be understood by non-technical audiences.

6.2 Evaluation & Reflections

Despite a change from the proposal, in the context of the research question and objectives, this project was successful. These differed in focus from those initially proposed as no automatically extracted data was available (and so it was not possible to imitate the format of this data). This meant one research question was dropped – this was "*Can a domain expert's understanding be combined with algorithmically extracted data from unstructured sources to yield knowledge graphs that model food policy networks more accurately and completely?*". This is likely to have been beneficial on two fronts:

1 The initially proposed pair of research questions were highly ambitious. It was unlikely that both would be achieved in the timescale of the project. A single research question enabled a greater level of focus, and so higher quality research.

2 The dropped research question is novel in its application domain, but standard practice among ontologists. The retained question offers novel contributions within both food policy and visualisation domains. This allowed for more impactful work.

This change could be accommodated due to the agile approaches, and iterative prototyping methodology. This flexible approach to development was invaluable, as it enabled the incorporation of substantial changes in approach. One such example is the incorporation of user study, which had not been planned at the outset and required construction of an interview protocol and a second set of participant forms. This approach also prevented the cessation of development at certain key points, as work could be directed elsewhere. The collaborating domain experts were on holiday when the ontology needed approval, and so sign-off was delayed. This time was used to develop the knowledge graph creation approach, and basic visualisation capability, saving time later in the development process. This did involve more stress on the part of the researcher until all of the parts were working together, and from accommodating the needs and requests of collaborators, but was ultimately worthwhile.

Communication was an unforeseen challenge of this project. The collaborating food policy experts had no prior experience of semantic web technologies, and the researcher no experience of food policies. There was a great deal of technical language and information that needed to be exchanged between parties. This is a common challenge in ontology development (Westerinen and Tauber, 2017), but presenting the proposed ontology and changes to the domain experts for approval was very difficult. Deciding which information not to present to avoid losing clarity was equally important. Some modelling choices needed to be explained, despite being obvious to an ontologist. One such example was why Person and Group are treated as disjoint subclasses of Actor, rather than Person being a subclass of Group. This is because a Person can be a member of a Group, but a Person is not a type of Group. This became even more challenging when developing inference rules, where they needed to be communicated and tested for validity in the domain through conversation. The researcher's prior experience in communicating complex ideas to novice audiences as a science teacher was invaluable on this front.

The challenges in this project led to significant professional development on the part of the researcher. The researcher had not previously developed anything on the scale of this project. While they had developed visualisations, they were far simpler in scope and had very limited interactive capabilities. Ontology and knowledge graph development had been done in isolation. No software had ever been developed for users. The visualisation tools were entirely unfamiliar. Some of the greatest challenges and accomplishments resulted in relatively insignificant outcomes in the scope of the project. The most notable of these was extracting the positions of all nodes from the visualisation, for precomputation purposes. This capability is not present in Dash-Cytoscape, and no prior solution or workaround existed. This was accomplished through modifying the source code of the package, which is written in javascript, a language unknown to the researcher. The outcome of this problem is relatively insignificant – a set of coordinates. However, this process of identifying an unsolved problem, conceiving of a solution, and working to achieve it was highly significant for the researcher as a marker of their own mastery of the discipline.

Research Objective 1 was achieved. An ontology was developed in collaboration with food policy experts. This was used to construct knowledge graph of the domain, which was refined through multiple iterations.

Research Objective 2 was achieved. An interactive knowledge graph visualisation tool was designed, implemented, and evaluated through user study. It was shown to support the reflective and reasoning process of domain expert users and was understood by non-technical audiences.

This work has presented answers to the Research Question. A domain knowledge graph of food policy has been visualised effectively. It has been shown, in a limited capacity, to support the reflective and reasoning process of domain experts. The principles and design approaches can be used to inform the development of other domain knowledge graph visualisations.

6.3 Conclusion

We have presented a domain specific ontology, knowledge graph, and visualisation software that were developed using a novel parallel approach. The food waste ontology was appropriate and generalised to represent other food policy experts' understanding of their field. The simple knowledge graph served its purpose in facilitating visualisation. The visualisation software has allowed its target userbase of domain experts with no prior semantic web experience to interface with the knowledge graph. Domain experts were able to reason, reflect and learn using the knowledge graph visualisation without any prior semantic web experience.

References

Abu-Salih, B. (2021) ‘Domain-specific knowledge graphs: A survey’, *Journal of Network and Computer Applications*, 185, p. 103076. doi:https://doi.org/10.1016/j.jnca.2021.103076.

Allemang, D. and Hendler, J.A. (2011) *Semantic Web for the working ontologist: effective modeling in RDFS and OWL*. 2nd edn. Waltham, MA: Morgan Kaufmann/Elsevier (Book, Whole). Available at: https://go.exlibris.link/ZRHKnjsS.

Andrienko, N. *et al.* (2018) ‘Viewing Visual Analytics as Model Building: Viewing Visual Analytics as Model Building’, *Computer Graphics Forum*, 37(6), pp. 275–299. doi:10.1111/cgf.13324.

Asmat, M.R.A., Wiens, V. and Lohmann, S. (2018) ‘A Comparative User Evaluation on Visual Ontology Modeling Using Node-Link Diagrams’, in *Emerging Topics in Semantic Technologies*. IOS Press, pp. 1–12.

Atemezing, G.A. and Troncy, R. (no date) ‘Towards a Linked-Data based Visualization Wizard.’, in.

Batista, F. *et al.* (2006) ‘Ontology construction: cooking domain’, *Artificial Intelligence: Methodology, Systems, and Applications*, 41(1), p. 30.

Bellini, P., Nesi, P. and Venturi, A. (2014) ‘Linked open graph: Browsing multiple SPARQL entry points to build your own LOD views’, *Journal of Visual Languages & Computing*, 25(6), pp. 703–716. doi:10.1016/j.jvlc.2014.10.003.

Bikakis, N. *et al.* (2016) ‘graphVizdb: A scalable platform for interactive large graph visualization’, in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, Helsinki, Finland: IEEE, pp. 1342–1345. doi:10.1109/ICDE.2016.7498340.

Boelsen-Robinson, T. *et al.* (2021) ‘Mapping factors associated with a successful shift towards healthier food retail in community-based organisations: A systems approach’, *Food Policy*, 101, p. 102032. doi:10.1016/j.foodpol.2021.102032.

Burch, M. *et al.* (2021) ‘The State of the Art in Empirical User Evaluation of Graph Visualizations’, *IEEE Access*, 9, pp. 4173–4198. doi:10.1109/ACCESS.2020.3047616.

Camarda, D.V., Mazzini, S. and Antonuccio, A. (2012) ‘LodLive, exploring the web of data’, in *Proceedings of the 8th International Conference on Semantic Systems - I-SEMANTICS ’12*. *the 8th International Conference*, Graz, Austria: ACM Press, p. 197. doi:10.1145/2362499.2362532.

Caracciolo, C. *et al.* (2013) 'The AGROVOC linked dataset', *Semantic Web*, 4(3), pp. 341–348.

Chawuthai, R. and Takeda, H. (2016) 'RDF Graph Visualization by Interpreting Linked Data as Knowledge', in Qi, G. et al. (eds) *Semantic Technology*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 23–39. doi:10.1007/978-3-319-31676-5_2.

Chen, J. *et al.* (2021) 'OWL2Vec*: embedding of OWL ontologies', *Machine Learning*, 110(7), pp. 1813–1845. doi:10.1007/s10994-021-05997-6.

Chen, Y. *et al.* (2019) 'AgriKG: An Agricultural Knowledge Graph and Its Applications', in Li, G. et al. (eds) *Database Systems for Advanced Applications*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 533–537. doi:10.1007/978-3-030-18590-9_81.

Dawson, C. (2015) *Projects in Computing and Information Systems 3rd Edn: a Student's Guide*. Harlow, United Kingdom: Pearson Education Canada. Available at: https://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=5832484 (Accessed: 27 April 2021).

Dogrusoz, U. *et al.* (2009) 'A layout algorithm for undirected compound graphs', *Information Sciences*, 179(7), pp. 980–994. doi:10.1016/j.ins.2008.11.017.

Dooley, D.M. *et al.* (2018) 'FoodOn: a harmonized food ontology to increase global food traceability, quality control and data integration', *npj Science of Food*, 2(1), p. 23. doi:10.1038/s41538-018-0032-6.

Dudáš, M. *et al.* (2018) 'Ontology visualization methods and tools: a survey of the state of the art', *The Knowledge Engineering Review*, 33.

Dwyer, T. *et al.* (2009) 'A Comparison of User-Generated and Automatic Graph Layouts', *IEEE Transactions on Visualization and Computer Graphics*, 15(6), pp. 961–968. doi:10.1109/TVCG.2009.109.

Endert, A. *et al.* (2014) 'The human is the loop: new directions for visual analytics', *Journal of Intelligent Information Systems*, 43(3), pp. 411–435. doi:10.1007/s10844-014-0304-9.

Ericksen, P.J. (2008) 'Conceptualizing food systems for global environmental change research', *Global Environmental Change*, 18(1), pp. 234–245. doi:10.1016/j.gloenvcha.2007.09.002.

Fanzo, J. *et al.* (2020) 'The Food Systems Dashboard is a new tool to inform better food policy', *Nature Food*, 1(5), pp. 243–246. doi:10.1038/s43016-020-0077-y.

Ferré, S. (2014) ‘Sparklis: A SPARQL Endpoint Explorer for Expressive Question Answering’, in *Proceedings of the 2014 International Conference on Posters & Demonstrations Track - Volume 1272*. Aachen, DEU: CEUR-WS.org (ISWC-PD’14), pp. 45–48.

Forbes, H., Quested, T. and O’Connor, Clementine (2021) *Food Waste Index Report 2021*. UNEP.

Franz, M. *et al.* (2015) ‘Cytoscape.js: a graph theory library for visualisation and analysis’, *Bioinformatics*, p. btv557. doi:10.1093/bioinformatics/btv557.

Gómez-Romero, J. and Molina-Solana, M. (2018) ‘GraphDL: An Ontology for Linked Data Visualization’, in Herrera, F. et al. (eds) *Advances in Artificial Intelligence*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 351–360. doi:10.1007/978-3-030-00374-6_33.

Haag, F. *et al.* (2015) ‘QueryVOWL: Visual Composition of SPARQL Queries’, in Gandon, F. et al. (eds) *The Semantic Web: ESWC 2015 Satellite Events*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 62–66. doi:10.1007/978-3-319-25639-9_12.

Harrower, M. and Brewer, C.A. (2003) ‘ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps’, *The Cartographic Journal*, 40(1), pp. 27–37. doi:10.1179/000870403235002042.

Haussmann, S. *et al.* (2019) ‘FoodKG: A Semantics-Driven Knowledge Graph for Food Recommendation’, in Ghidini, C. et al. (eds) *The Semantic Web – ISWC 2019*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 146–162. doi:10.1007/978-3-030-30796-7_10.

Hitzler, P., Krotzsch, M. and Rudolph, S. (2009) *Foundations of semantic web technologies*. Chapman and Hall/CRC.

HL Deb. vol. 811 col. 816-819, 24 March 2021 (2021). Hansard. Available at: https://hansard.parliament.uk/lords/2021-03-24/debates/922AFE55-9B0B-4500-9216-DA60B0A13E5C/FoodWaste (Accessed: 1 September 2021).

Hogan, A. *et al.* (2021) ‘Knowledge Graphs’, *arXiv:2003.02320 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2003.02320 (Accessed: 25 April 2021).

Holten, D. and van Wijk, J.J. (2009) ‘A user study on visualizing directed edges in graphs’, in *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09. the SIGCHI Conference*, Boston, MA, USA: ACM Press, p. 2299. doi:10.1145/1518701.1519054.

Jordan, P.W. *et al.* (eds) (1996) 'SUS: A "Quick and Dirty" Usability Scale', in *Usability Evaluation In Industry*. CRC Press, pp. 207–212. doi:10.1201/9781498710411-35.

Kiambi, S. *et al.* (2018) 'Mapping Nairobi's dairy food system: An essential analysis for policy, industry and research', *Agricultural Systems*, 167, pp. 47–60. doi:10.1016/j.agsy.2018.08.007.

Kolchin, M. and Zamula, D. (2013) 'Food product ontology: Initial implementation of a vocabulary for describing food products', in *Proceeding of the 14th Conference of Open Innovations Association FRUCT, Helsinki, Finland*, pp. 11–15.

Kurteva, A. and De Ribaupierre, H. (2021) 'Interface to Query and Visualise Definitions from a Knowledge Base', *arXiv:2103.06571 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2103.06571 (Accessed: 16 April 2021).

Levkoe, C.Z. *et al.* (2021) 'Mapping Food Policy Groups: Understanding Cross-Sectoral Network Building through Social Network Analysis', *Canadian Food Studies / La Revue canadienne des études sur l'alimentation*, 8(2). doi:10.15353/cfs-rcea.v8i2.443.

Lohmann, S. *et al.* (2010) 'The RelFinder user interface: interactive exploration of relationships between objects of interest', in *Proceedings of the 15th international conference on Intelligent user interfaces - IUI '10*. *the 15th international conference*, Hong Kong, China: ACM Press, p. 421. doi:10.1145/1719970.1720052.

Lohmann, S. *et al.* (2015) 'WebVOWL: Web-based Visualization of Ontologies', in *Proceedings of EKAW 2014 Satellite Events*. Springer (LNAI), pp. 154–158.

Martins, F. *et al.* (2008) 'Starting to cook a tutoring dialogue system', in *2008 IEEE Spoken Language Technology Workshop*. *2008 IEEE Spoken Language Technology Workshop (SLT)*, Goa, India: IEEE, pp. 145–148. doi:10.1109/SLT.2008.4777861.

Mialon, M., Crosbie, E. and Sacks, G. (2020) 'Mapping of food industry strategies to influence public health policy, research and practice in South Africa', *International Journal of Public Health*, 65(7), pp. 1027–1036. doi:10.1007/s00038-020-01407-1.

Micsik, A., Turbucz, S. and Tóth, Z. (2015) 'Exploring publication metadata graphs with the LODmilla browser and editor', *International Journal on Digital Libraries*, 16(1), pp. 15–24. doi:10.1007/s00799-014-0130-2.

Mikolov, T. *et al.* (2013) 'Efficient Estimation of Word Representations in Vector Space', *arXiv:1301.3781 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1301.3781 (Accessed: 15 September 2021).

Munzner, T. (2009) 'A Nested Model for Visualization Design and Validation', *IEEE Transactions on Visualization and Computer Graphics*, 15(6), pp. 921–928. doi:10.1109/TVCG.2009.111.

Musen, M.A. (2015a) ‘The protégé project: a look back and a look forward’, *AI Matters*, 1(4), pp. 4–12. doi:10.1145/2757001.2757003.

Musen, M.A. (2015b) ‘The Protégé Project: A Look Back and a Look Forward.’, *AI matters*, 1(4), pp. 4–12. doi:10.1145/2757001.2757003.

Oates, B.J. (2006) *Researching information systems and computing*. Available at: http://0-search.ebscohost.com.cataleg.uoc.edu/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1099431 (Accessed: 26 April 2021).

Our Waste, Our Resources: A Strategy For England (2018). DEFRA.

Parsons, K. (2020) *Who Makes Food Policy in England?: A Map of Government Actors and Activities*. Food Research Collaboration (Rethinking Food Governance).

Parsons, K., Barling, D. and Lang, T. (2018) ‘Chapter Seven - UK Policymaking Institutions and Their Implications for Integrated Food Policy’, in Barling, D. and Fanzo, J. (eds). Elsevier (Advances in Food Security and Sustainability), pp. 211–251. doi:10.1016/bs.af2s.2018.09.005.

Paulheim, H. (2016) ‘Knowledge graph refinement: A survey of approaches and evaluation methods’, *Semantic Web*. Edited by P. Cimiano, 8(3), pp. 489–508. doi:10.3233/SW-160218.

Pesquita, C. *et al.* (2018) ‘A framework to conduct and report on empirical user studies in semantic web contexts’, in *European Knowledge Acquisition Workshop*. Springer, pp. 567–583.

Po, L. *et al.* (2020) ‘Linked Data Visualization: Techniques, Tools, and Big Data’, *Synthesis Lectures on the Semantic Web: Theory and Technology*, 10(1), pp. 1–157. doi:10.2200/S00967ED1V01Y201911WBE019.

Po, L. and Malvezzi, D. (no date) ‘High-level Visualization Over Big Linked Data.’, in.

Po, L. and Papastefanatos, G. (2020) ‘A Comparative Study of State-of-The-Art Linked Data Visualization Tools’.

Qin, L., Hao, Z. and Zhao, L. (2019) ‘Food safety Knowledge Graph and Question Answering System’, in *Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City*. *ICIT 2019: IoT and Smart City*, Shanghai China: ACM, pp. 559–564. doi:10.1145/3377170.3377260.

Ren, D. *et al.* (2019) ‘Understanding node-link and matrix visualizations of networks: A large-scale online experiment’, *Network Science*, 7(2), pp. 242–264. doi:10.1017/nws.2019.6.

Reynolds, C. *et al.* (2019) ‘Review: Consumption-stage food waste reduction interventions – What works and how to design better interventions’, *Food Policy*, 83, pp. 7–27. doi:10.1016/j.foodpol.2019.01.009.

Savary, S. *et al.* (2020) ‘Mapping disruption and resilience mechanisms in food systems’, *Food Security*, 12(4), pp. 695–717. doi:10.1007/s12571-020-01093-0.

Schlegel, K. *et al.* (2014) ‘Balloon Synopsis: A Modern Node-Centric RDF Viewer and Browser for the Web’, in Presutti, V. et al. (eds) *The Semantic Web: ESWC 2014 Satellite Events*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 249–253. doi:10.1007/978-3-319-11955-7_29.

Shimizu, C. and Hammar, K. (2019) ‘CoModIDE - The Comprehensive Modular Ontology IDE’, in *18th International Semantic Web Conference: Satellite Events*.

Shirai, S.S. *et al.* (2021) ‘Identifying Ingredient Substitutions Using a Knowledge Graph of Food’, *Frontiers in Artificial Intelligence*, 3, p. 621766. doi:10.3389/frai.2020.621766.

Shneiderman, B. (1996) ‘The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations’, in *Proceedings of the 1996 IEEE Symposium on Visual Languages*. USA: IEEE Computer Society (VL ’96), p. 336.

Sirin, E. *et al.* (2007) ‘Pellet: A practical OWL-DL reasoner’, *Journal of Web Semantics*, 5(2), pp. 51–53. doi:10.1016/j.websem.2007.03.004.

Soylu, A. *et al.* (2018) ‘OptiqueVQS: A visual query system over ontologies for industry’, *Semantic Web*. Edited by F. Lecue, 9(5), pp. 627–660. doi:10.3233/SW-180293.

Takezaki, A. *et al.* (2020) ‘Common Agriculture Vocabulary for Enhancing Semantic-level Interoperability in Japan’, *Japan Agricultural Research Quarterly: JARQ*, 54(3), pp. 219–225.

Tomaszuk, D. and Truchan, P. (2017) ‘VisGraph^ 3: a web tool for RDF visualization and creation’.

Vega-Gorgojo, G. *et al.* (2019) ‘Linked Data Exploration With RDF Surveyor’, *IEEE Access*, 7, pp. 172199–172213. doi:10.1109/ACCESS.2019.2956345.

Viola, F. *et al.* (2018) ‘Interactive 3D Exploration of RDF Graphs through Semantic Planes’, *Future Internet*, 10(8), p. 81. doi:10.3390/fi10080081.

Vrandečić, D. and Krötzsch, M. (2014) ‘Wikidata: a free collaborative knowledgebase’, *Communications of the ACM*, 57(10), pp. 78–85. doi:10.1145/2629489.

Ware, C. and Bobrow, R. (2005) ‘Supporting visual queries on medium-sized node–link diagrams’, *Information Visualization*, 4(1), pp. 49–58.

Weise, M., Lohmann, S. and Haag, F. (2016) ‘Ld-vowl: Extracting and visualizing schema information for linked data’, in *2nd international workshop on visualization and interaction for ontologies and linked data*, pp. 120–127.

Westerinen, A. and Tauber, R. (2017) ‘Ontology development by domain experts (without using the “O” word)’, *Applied Ontology*. Edited by K. Baclawski and M. Bennett, 12(3–4), pp. 299–311. doi:10.3233/AO-170183.

Yoghourdjian, V. *et al.* (2018) ‘Exploring the limits of complexity: A survey of empirical studies on graph visualisation’, *Visual Informatics*, 2(4), pp. 264–282. doi:10.1016/j.visinf.2018.12.006.

Zaveri, A. *et al.* (2013) ‘User-Driven Quality Evaluation of DBpedia’, in *Proceedings of the 9th International Conference on Semantic Systems*. New York, NY, USA: Association for Computing Machinery (I-SEMANTICS ’13), pp. 97–104. doi:10.1145/2506182.2506195.

Zaveri, A. *et al.* (2015) ‘Quality assessment for Linked Data: A Survey: A systematic literature review and conceptual framework’, *Semantic Web*. Edited by P. Hitzler, 7(1), pp. 63–93. doi:10.3233/SW-150175.

Zulaika, U., Gutiérrez, A. and López-de-Ipiña, D. (2018) ‘Enhancing Profile and Context Aware Relevant Food Search through Knowledge Graphs’, *Proceedings*, 2(19), p. 1228. doi:10.3390/proceedings2191228.

Appendices

Appendix A Project Proposal

Research Proposal ***Ollie Keers***

adbr309

Knowledge Graph of Food Policy Networks: Construction, Visualisation & Refinement

Introduction

The project aims to determine whether knowledge graph visualisations can aid in the reflective and reasoning process of domain experts, using the domain of food policy as a case study. Traditional networks capture the existence of relationships between entities but do not describe either the nature of entities or their relationships. Creating a domain-specific ontology and using this to transform simple messy network data into a knowledge graph has the potential to enrich the data and improve its utility. This algorithmically generated knowledge graph would likely still contain inaccuracies, so interactive visualisation of the knowledge graph will allow for domain-expert review and improvement. These visualisations could both domain experts and wider audiences to develop a better understanding of the complex and interconnected systems within the food policy domain.

Food policies are important because beyond politics they affect public health, the economy, and society at large. Food policy networks are conceptual models of the complex interconnections between policies, policymakers, departments and stakeholders. Understanding food policy networks is fundamental to effect changes, but unconsolidated data makes research time-consuming and often leads to incomplete understanding. Ongoing work aims to capture these networks more completely through automated data extraction from unstructured sources, but these may contain inaccuracies. Knowledge graphs offer the opportunity to benefit from both the accuracy of expert knowledge and the completeness of algorithmic generation. Visualisations of knowledge graphs could support food policy experts in reasoning and reflecting on the domain, and other individuals in acquiring knowledge of the domain.

In this context the project seeks to answer the following 2 research questions:

RQ1: Can a domain expert's understanding be combined with algorithmically extracted data from unstructured sources to yield knowledge graphs that model food policy networks more accurately and completely?

RQ2: How can knowledge graphs of food policy networks be visualised effectively to support the reflective and reasoning process of domain experts, and to communicate this to wider audiences?

To answer these questions the project will pursue the following 2 objectives:

RO1: To improve the accuracy and completeness of automatically extracted food policy networks by developing an ontology in collaboration with food policy experts and using this to construct and refine a knowledge graph of the domain.

RO2: To design, implement and evaluate interactive knowledge graph visualisations that effectively support the reflective and reasoning process of domain expert users, and can be understood by non-technical audiences.

Knowledge graphs aim to model conceptual networks, but algorithmic generation leads to quality issues. Refinement of knowledge graphs requires both significant technical knowledge and domain expertise, often necessitating collaboration. Visualisation of knowledge graphs is uncommon due to the challenges posed by the range of information they contain but has the potential to improve accessibility. Development of knowledge graph visualisations that can be interactively refined by non-technical domain experts has the potential to broaden the impact of this technology.

Critical Context

Food policies are important because beyond politics they affect public health, the economy, and society at large. Food policies cover any policy that has an impact upon food systems. These can be directly focused on food, such as Free School Meals for deprived children, or indirectly have influence it, such as supply issues due to increased paperwork from Brexit. Food policy can have wide-ranging effects including the diets of individuals, range of choice available, and the profitability of products.

Food policy networks are conceptual models of the complex interconnections between policies, policymakers, departments and stakeholders. Food is fundamental to survival, but the systems governing it are complex (Fanzo *et al.*, 2020). Food policy networks are conceptual mappings of all of the entities and relationships within the food system. These entities include various government departments, pieces of legislation, corporations, charities, consumers and producers. The relationships between these are directed in nature, and include collaborating, competing, taxing and shopping.

Understanding food policy networks is fundamental to affect changes, but unconsolidated data makes research time-consuming and often leads to incomplete understanding. Food policy within the UK is fragmented and has been assembled by multiple governments and different departments over the years (Parsons, Barling and Lang, 2018). There is no unified collection of data on food policies, with information in multiple sources such as policy documents, parliamentary minutes and press releases. Researching this area requires searching for all relevant documents and reading a large corpus of text. This is an inefficient and likely incomplete process. Documents may be self-contained, and so the links between them are not always apparent. The user is required to infer many relationships between entities – a process that may be limited by their own understanding of the domain. The dynamic nature of this constantly evolving domain also makes it challenging for users to acquire and maintain a comprehensive understanding of the networks. These factors suggest that an algorithmic, data driven approach to modelling these networks would be useful.

Knowledge graphs offer the opportunity to benefit from both the accuracy of expert knowledge and the completeness of algorithmic generation. Algorithmically generated food policy networks may contain inaccuracies and lack detail. Expert-assembled food policy networks may not be comprehensive or up to date with recent developments. A knowledge graph in general terms is a "graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities" (Hogan *et al.*, 2021). Knowledge graphs are typically constructed by applying domain-expert curated rules and schema to algorithmically extracted entities and relationships, benefitting from the strengths of each. Domain knowledge graphs aim to semantically model all the relationships between entities within that specific domain (Abu-Salih, 2021). This makes a domain knowledge graph well-suited for the purposes of representing a food policy network.

Knowledge graphs aim to model conceptual networks, but algorithmic generation leads to quality issues. The quality of a knowledge graph can be judged by many metrics, which

beyond accuracy and completeness also includes conciseness (Zaveri *et al.*, 2015). The underlying data used to create the knowledge graph is unlikely to comprehensively model the domain and may also contain errors, inconsistencies or irrelevant information. Knowledge extraction and graph construction are probabilistic or rule-based processes, and so will introduce errors. Roughly 12% of the relationships on DBpedia, one of the largest knowledge graphs, were found to have quality issues (Zaveri *et al.*, 2013). Capturing all relevant relationships correctly without introducing any incorrect or irrelevant assertions is challenging if not impossible for all but the smallest of knowledge graphs.

Refinement of knowledge graphs requires both significant technical knowledge and domain expertise, often necessitating collaboration. Knowledge graph refinement consists of both identification and removal of errors, and completion by adding missing assertions (Paulheim, 2016). Execution of these tasks can require an understanding of symbolic logic, probabilistic machine learning and natural language processing. Knowledge graphs are abstract conceptual networks of relationships stored as text, making them unfriendly for non-specialists to use. Domain-specific knowledge graphs aim to model specific concepts that require an intricate knowledge of the domain, which is not likely to be held by a knowledge graph specialist. Producing a high-quality knowledge graph therefore requires collaboration between the domain expert and programmer.

Visualisation of knowledge graphs is uncommon due to the challenges posed by the range of information they contain but has the potential to improve accessibility. Knowledge graphs are very large, with Google's Knowledge graph consisting of 18 billion facts across 1500 different types of entity (Paulheim, 2016). Both the scale and range of information lead to significant complexity in any visualisation, limiting their utility. However, effective visualisation of the knowledge graph would enable a human to interface with it and build their mental model of the systems that it represents (Andrienko *et al.*, 2018). Visualisation of ontologies, abstractions of relations without specific instances, is more common. A recent survey of ontology visualisation tools (Dudáš *et al.*, 2018) aimed at expert users indicated that current approaches leave much to be desired, even at this level.

Visualisations of knowledge graphs could support food policy experts in reasoning and reflecting on the domain, and other individuals in acquiring knowledge of the domain. Visualisation allows domain-experts to interface with the knowledge graph. Viewing knowledge graph visualisations has the potential to contribute to the domain expert's knowledge discovery process, or to reinforce prior knowledge. Visualisation also allows for the knowledge contained within the graphs to be shared more widely with novice audiences.

Development of knowledge graph visualisations that can be interactively refined by non-technical domain experts has the potential to broaden the impact of this technology. Recent work developed a user-interface to allow novice users to query and visualise definitions from DBpedia (Kurteva and De Ribaupierre, 2021). While the level of interactivity was somewhat limited, this was found to be helpful and easy to use, with highly educated participants stating that they would use the tool frequently, demonstrating the value of knowledge graph visualisations. Interactive visualisations of domain-specific knowledge graphs could support domain-specialists in using them without needing to learn the

technologies that underpin them. A wider domain-expert userbase can better provide feedback to refine knowledge graphs, increasing their utility.

Approaches: Methods & Tools for Design, Analysis & Evaluation

Overview of Methodology:

A food policy ontology will be created with experts. A dictionary of relevant entities and terms, classified by domain-specialists, will be used for automated recognition of entities in pseudo-scraped data. This allows for alignment of the data with the ontology, and knowledge graph building using logical reasoning. The knowledge graph will be visualised. Both the knowledge graph and the visualisations will be iteratively evaluated and refined using human-in-the-loop approaches, consulting the domain expert as appropriate.

General Information

This work will take a design & create approach, following an iterative prototyping methodology (Oates, 2006). The project aims to establish the extent to which it is possible to create and visualise a model of a collaborating domain expert's mental model of food policy networks, using a case study of food waste. This process will involve iterative refinement following feedback from the domain expert on the quality of both the knowledge graph and the interactive visualisations thereof.

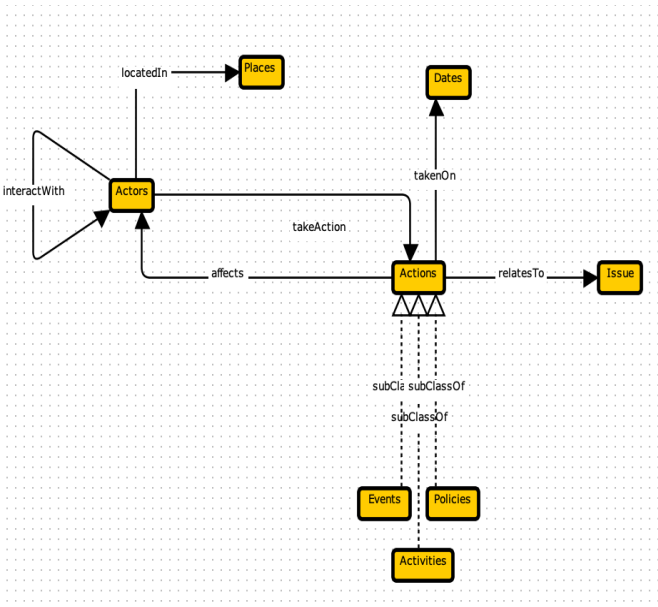
A literature review will be performed, expanding upon that already undertaken in construction of this proposal. This will largely focus on network visualisation approaches, with some work on knowledge graph generation and refinement. This will not include research into the domain of food policy, as this knowledge and understanding is to come from the collaborating domain expert.

The data used for this project will be human assembled pseudo-scraped data, from collaborators at Wren & Co, a food policy consultancy. This project is intended to follow from the acquisition of food policy data from scraping publicly available documents, so the data used here will emulate the output of the scraping process. This knowledge graph visualisation project aims to eventually be applicable to the scraped dataset.

Ontology & Knowledge Graph Construction

An ontology is being developed in collaboration with the domain expert to represent the classes of individuals and their interactions within this domain. Ontologies are domain-specific categorizations of the classes of entities, their properties, and the relationships between them. These are formalized rules describing what is conceptually permissible within that domain. Ontologies are domain-specific: an actor in general use is someone who performs in a play, but within the food policy domain an actor any individual or organization who interacts with the network. Ontologies may contain restrictions on properties and interactions (corporations pay tax to government, but never the inverse), or otherwise contain rules that allow reasoning to take place (such as the transitive property 'located in'). Ontologies are created separately from data, which is used later to create the knowledge graph.

Protégé is being used to develop the ontology. Protégé (Musen, 2015) is an ontology development tool with a graphical user interface. It also contains reasoners (software to apply logic to the ontology) and visualisation tools. The visualisation tools are of limited use for directly editing the ontology, but they are useful for showing the ontology to non-technical domain experts. Preliminary work to construct a simple ontology has been performed through observation of the domain expert performing a text extraction and categorization task. A visualisation of this, produced with CoModIDE (Shimizu and Hammar, 2019) is shown to the right. Expert feedback suggests that this sufficiently models the domain in broad terms, but it is lacking detail and will need further refinement as the project progresses.



A dictionary of all significant entities and terms within the food policy domain, provided by Wren & Co, will be used for entity recognition in the pseudo-scraped dataset. The pseudo-scraped dataset does not assign individual data to the classes defined in the ontology. To determine the nature of entities, term recognition will be used based upon lexical similarity and pattern matching. Terms will be compared with a human-written dictionary of significant entities within the food policy domain. This allows for entities to be assigned as government departments, pieces of legislation, dates or other items of interest in subsequent steps.

A knowledge graph modelling the domain in detail will be constructed by combining the pseudo-scraped dataset with the expert-curated ontology and logical reasoning. The pseudo-scraped dataset establishes that there exist relationships of an unknown nature between unclassified entities, such as between the Department for Education (DfE) and Free School Meals (FSM). Entities within the pseudo-scraped dataset are assigned to classes through recognition, such as that the DfE is a Government Department, and FSM is a Policy. The ontology establishes rules that govern the directed relationships between classes within the domain, such as that government departments write policies. Logical reasoning can be used to infer the nature of this relationship, that the DfE wrote FSM policy. These are added to the knowledge graph using a package such as **rdflib**. A reasoner can then be used, either in a package like **owlready2** or in **Protégé**, to infer further relationships and check for logical inconsistencies.

The knowledge graph is a series of directed, three-item statements known as triples. Triples are a three-item statement, of the format: subject, predicate, object. The subject and object are nodes, with the predicate being a description of the directed relationship from subject to object. Triples, or combinations of triples, can be used to describe most, but not all, statements and relationships. Resource Descriptive Framework (**RDF**) is the W3C standard for representing triples and will be adopted in the project for representing the graph.

The knowledge graph will need multiple iterations of refinement. The constructed knowledge graph will neither capture all relationships that exist, nor will all of the constructed triples represent real relationships. Errors will exist in the knowledge graph, with its improvement being an iterative process. Improvement of the knowledge graph can involve adjustment of the creation process to incorporate new rules or recognized terms, adjustment of the ontology, or manual creation of triples as appropriate. Some of the errors may be trivially identifiable without expert domain knowledge, while others will require inspection by the collaborating expert. Providing an accessible way for the domain expert to inspect a knowledge graph requires its visualisation.

The knowledge graph will be constructed under the Open World Assumption (OWA). As the food policy domain is not static OWA will be adopted, indicating that the knowledge graph is not a complete description of the domain. Missing entities or relationships do not indicate that they do not exist in the real world. Further information can be added to the graph, through the addition of further RDF triples. In this way, the graph can be refined and developed.

Knowledge Graph Visualisation

The initial visualisation of the knowledge graph will guide the development of subsequent improvements. The challenges of making network visualisations usable are well documented (Yoghourdjian *et al.*, 2018; Burch *et al.*, 2021). Representing **node-link diagrams** in a manner that allows the user to extract understanding is a tradeoff between complexity and comprehensibility. Above the complexity of a standard network, this knowledge graph will contain directed edges, edge labels, and errors from the automated graph creation process. An initial visualisation of the network will be produced to identify areas for improvement. Establishing how these challenges are best met will be an iterative process of development and review.

Improvements to this initial visualisation will be needed to support interpretation, which are likely to include representation of the directed edges, force-directed edges, and user-interactivity. The requirements of the domain expert are to be determined after viewing early visualisations. It is possible to assume some desirable functionality based upon the literature (Burch *et al.*, 2021). Directed edges are required, which may be more suitable represented as arrows or tapered lines (Holten and van Wijk, 2009). Edge labels are required, with the potential for these to be encoded as static text or otherwise such as through the use of colours or tooltips. Either force-directed or user-generated layouts may be suitable(Dwyer *et al.*, 2009), and in this case the latter may well be more useful for the task of representing that same user's mental model.

Given the knowledge graph is likely to consist of several thousand nodes, filtering is likely to be a useful interaction which may be implemented through SPARQL querying. Visualisation of the entire network may not provide any substantial value to the user, besides demonstrating that food policy is a complex and interrelated network. It is likely that interactivity in line with the information seeking mantra of 'overview first, zoom & filter, then details on demand" (Shneiderman, 1996) will be useful. Filtering of knowledge graphs is typically achieved through SPARQL queries, and it is likely that these will be employed here.

The output can be processed to generate an appropriate format for subsequent visualisation such as through the use of GraphDL (Gómez-Romero and Molina-Solana, 2018), or to preserve the information such as in a csv file. Examples include all government departments that wrote free school meals policy, or all information within two edges of supermarkets. Such smaller parts of the knowledge graph are easier to visualise given their size and complexity, and more likely to enable the user to extract the information they require.

Visualisation tools will be selected based on the functionality they offer compared to the user requirements. It would not be practical to develop a novel visualisation package within the timescale of this project in addition to the knowledge graph creation process. Therefore, it is advisable to use an existing package that offers as much of the functionality required as possible. Initial research has suggested several suitable candidates, including **d3**, **pyvis**, **dash-cytoscape**, and **gephi**. The choice will be influenced by the functionality required by the user, which may evolve through the development process.

The knowledge graph and the visualisations will be refined iteratively from inspection of the visualisation. A typical visual analytics human-in-the-loop workflow will be employed for this iterative development process. In many cases it will be sufficient for the programmer to evaluate both the knowledge graph and visualisations directly, without need to consult the domain expert. To assess the quality of the knowledge graph, the evaluative framework proposed by Zaveri *et al.*, 2015 is useful. For the visualisation, this will be on the basis of whether it supports the user in extracting information. Once a visualisation of the knowledge graph has been deemed sufficiently usable, it will be shown to the domain expert for a more substantial evaluation as part of the iterative development process.

Evaluation

RQ1: Can a domain expert's understanding be combined with algorithmically extracted data from unstructured sources to yield knowledge graphs that model food policy networks more accurately and completely?

RQ1 will be evaluated through interviews with the domain expert following presentation of the final visualisations of the knowledge graph. These will explore the extent to which their mental model of the domain has been modelled faithfully.

RQ2: How can knowledge graphs of food policy networks be visualised effectively to support the reflective and reasoning process of domain experts, and to communicate this to wider audiences?

RQ2 will be evaluated through a critical review of the final visualisation in consultation with the domain expert against how successful it has been in conveying their mental model. This evaluation will include analysis of which aspects of the visualisation had contributed to its successes.

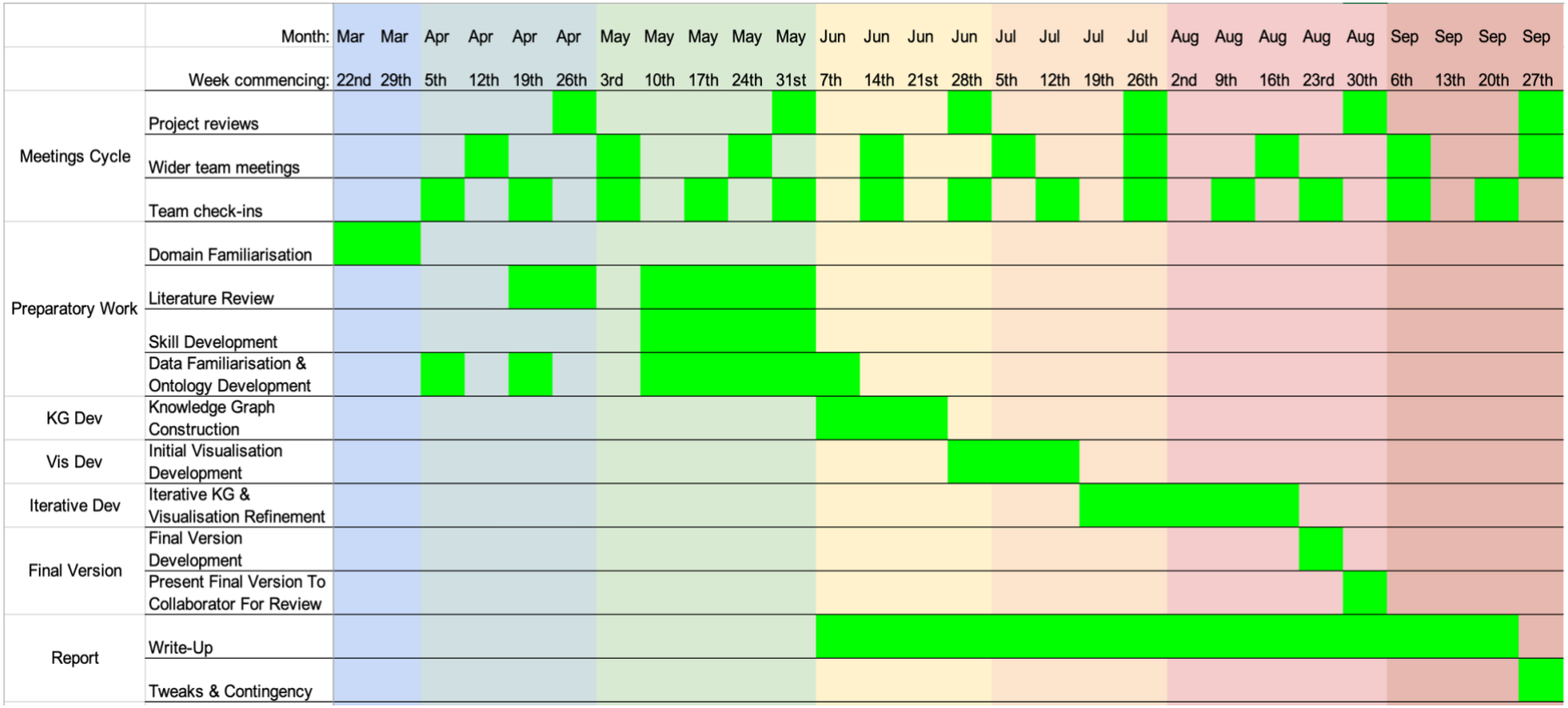
General Issues

This case study has the potential to be generalised to other domains. This work is a case study, using the domain of food policy networks, specifically food waste. It aims to model and

visualise the mental model of a particular domain expert and is therefore limited in its reproducibility. If the knowledge graph generation process is successful, it could be expanded to cover a greater domain or applied to other domains. Approaches for generating visualisations of knowledge graphs could be used on other, pre-existing knowledge graphs.

There are a number of potential research contributions from the work. Knowledge graph visualisation has had relatively little research. The application of a visual analytics human-in-the-loop workflow to knowledge graph refinement is also, as far as the author is aware, novel. In this way, both the tools to be developed and the process undertaken have the potential to enrich research.

There are no notable legal, professional & ethical issues that arise from this work. This work is undertaken as part of a collaboration between the GI Centre, City, University of London, and Wren & Co. The data provided by Wren & Co has been extracted from publicly available documents. No confidentiality issues exist. While this study is a collaboration with domain experts, it does not involve any human participants, and so no significant ethical issues were determined (see included Ethics Form). This work is unlikely to adversely affect the emotional, physical or mental wellbeing of anyone, or do any harm.



Notes

Work Plan

Additional Supervisor Meetings	These will be arranged in an ad-hoc fashion. Dr. Jianu to be kept abreast of development throughout vis email. Additional meetings likely to be required at the end of the initial visualisation development, and
Collaborator Reviews	These may take place either during the team check-in meetings or be arranged as additional sessions as appropriate.
Literature Review	To consist of both a thorough evaluation of both research papers and interactive network visualisation tools
Skill Development	This process will mainly be focused on network visualisation skills, as identified during the literature review

Risk

Using the impact assessment framework given by Dawson, 2015:

Risk	Likelihood (1-3)	Consequence (1-5)	Risk Impact (LxC)	Mitigation
Data/work loss due to file corruption etc.	1	5	5	Avoidance: Frequent backup to github & external hard-drive
Insufficient technical competence, limiting the scope of the work	2	3	6	Deflection: Project scope already adjusted to incorporate greater focus on knowledge graphs Avoidance: Time allocated to learn new technologies Contingency: Multiple technologies scoped, including more familiar or easier to use ones. Deflection: Seek guidance from supervisor
Data cannot be automatically transformed into a clean and usable KG	3	2	6	Contingency: Use more structured dataset for KG generation Contingency: Adjust scope and focus on visualisation Contingency: Accept this failure as a valid result with appropriate analysis as to the underlying causes
KG cannot be visualised appropriately with available technologies	2	3	6	Contingency: Use a selected subset of the KG for visualisation purposes, which may have been human-edited for suitability. Deflection: Multiple technologies evaluated early to understand the abilities and limitations of each and to enable switch if needed. Contingency: Any visualisations produced can be evaluated for effectiveness, with valid lessons to be taken from this regardless.
Unforeseen personal circumstances for self or collaborators interrupting work plan	2	4	8	Contingency: Early start to work planned and week of contingency at the end. Deflection: If collaborator unavailable, use another colleague from Wren & Co.
Collaborator withdrawal	1	5	5	Deflection: Formal agreement between GI centre & Wren & Co exists – this issue would be dealt with by more senior individuals Contingency: Apply these approaches to another dataset from a domain that I am familiar with, removing need for collaboration
Insufficient time allocated for task completion	1	2	2	Avoidance: Prototyping development process allows for stopping mid-way through development cycle if needed. First version planned to be completed by mid-July

Loss of motivation / burnout	1	5	5	Avoidance: Topic and tools chosen within areas that interest me. High work-ethic historically means that this is not a concern. Contingency: Work-life balance to be maintained throughout process. Work-plan to include sufficient time to avoid deadline crunch

References

Abu-Salih, B. (2021) 'Domain-specific knowledge graphs: A survey', *Journal of Network and Computer Applications*, 185, p. 103076. doi: https://doi.org/10.1016/j.jnca.2021.103076.

Andrienko, N. *et al.* (2018) 'Viewing Visual Analytics as Model Building: Viewing Visual Analytics as Model Building', *Computer Graphics Forum*, 37(6), pp. 275–299. doi: 10.1111/cgf.13324.

Burch, M. *et al.* (2021) 'The State of the Art in Empirical User Evaluation of Graph Visualisations', *IEEE Access*, 9, pp. 4173–4198. doi: 10.1109/ACCESS.2020.3047616.

Dawson, C. (2015) *Projects in Computing and Information Systems 3rd Edn: a Student's Guide*. Harlow, United Kingdom: Pearson Education Canada. Available at: https://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=5832484 (Accessed: 27 April 2021).

Dudáš, M. *et al.* (2018) 'Ontology visualisation methods and tools: a survey of the state of the art', *The Knowledge Engineering Review*, 33.

Dwyer, T. *et al.* (2009) 'A Comparison of User-Generated and Automatic Graph Layouts', *IEEE Transactions on Visualisation and Computer Graphics*, 15(6), pp. 961–968. doi: 10.1109/TVCG.2009.109.

Fanzo, J. *et al.* (2020) 'The Food Systems Dashboard is a new tool to inform better food policy', *Nature Food*, 1(5), pp. 243–246. doi: 10.1038/s43016-020-0077-y.

Gómez-Romero, J. and Molina-Solana, M. (2018) 'GraphDL: An Ontology for Linked Data Visualisation', in Herrera, F. et al. (eds) *Advances in Artificial Intelligence*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 351–360. doi: 10.1007/978-3-030-00374-6_33.

Hogan, A. *et al.* (2021) 'Knowledge Graphs', *arXiv:2003.02320 [cs]*. Available at: http://arxiv.org/abs/2003.02320 (Accessed: 25 April 2021).

Holten, D. and van Wijk, J. J. (2009) 'A user study on visualising directed edges in graphs', in *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*. *the SIGCHI Conference*, Boston, MA, USA: ACM Press, p. 2299. doi: 10.1145/1518701.1519054.

Kurteva, A. and De Ribaupierre, H. (2021) 'Interface to Query and Visualise Definitions from a Knowledge Base', *arXiv:2103.06571 [cs]*. Available at: http://arxiv.org/abs/2103.06571 (Accessed: 16 April 2021).

Musen, M. A. (2015) ‘The Protégé Project: A Look Back and a Look Forward.’, *AI matters*, 1(4), pp. 4–12. doi: 10.1145/2757001.2757003.

Oates, B. J. (2006) *Researching information systems and computing*. Available at: http://0-search.ebscohost.com.cataleg.uoc.edu/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1099431 (Accessed: 26 April 2021).

Parsons, K., Barling, D. and Lang, T. (2018) ‘Chapter Seven - UK Policymaking Institutions and Their Implications for Integrated Food Policy’, in Barling, D. and Fanzo, J. (eds). Elsevier (Advances in Food Security and Sustainability), pp. 211–251. doi: 10.1016/bs.af2s.2018.09.005.

Paulheim, H. (2016) ‘Knowledge graph refinement: A survey of approaches and evaluation methods’, *Semantic Web*. Edited by P. Cimiano, 8(3), pp. 489–508. doi: 10.3233/SW-160218.

Shimizu, C. and Hammar, K. (2019) ‘CoModIDE - The Comprehensive Modular Ontology IDE’, in *18th International Semantic Web Conference: Satellite Events*.

Shneiderman, B. (1996) ‘The Eyes Have It: A Task by Data Type Taxonomy for Information Visualisations’, in *Proceedings of the 1996 IEEE Symposium on Visual Languages*. USA: IEEE Computer Society (VL ’96), p. 336.

Yoghourdjian, V. *et al.* (2018) ‘Exploring the limits of complexity: A survey of empirical studies on graph visualisation’, *Visual Informatics*, 2(4), pp. 264–282. doi: 10.1016/j.visinf.2018.12.006.

Zaveri, A. *et al.* (2013) ‘User-Driven Quality Evaluation of DBpedia’, in *Proceedings of the 9th International Conference on Semantic Systems*. New York, NY, USA: Association for Computing Machinery (I-SEMANTICS ’13), pp. 97–104. doi: 10.1145/2506182.2506195.

Zaveri, A. *et al.* (2015) ‘Quality assessment for Linked Data: A Survey: A systematic literature review and conceptual framework’, *Semantic Web*. Edited by P. Hitzler, 7(1), pp. 63–93. doi: 10.3233/SW-150175.

Research Ethics Review Form: BSc, MSc and MA Projects

Computer Science Research Ethics Committee (CSREC)

http://www.city.ac.uk/department-computer-science/research-ethics

Undergraduate and postgraduate students undertaking their final project in the Department of Computer Science are required to consider the ethics of their project work and to ensure that it complies with research ethics guidelines. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people ("participants") in the project.

In order to ensure that appropriate consideration is given to ethical issues, all students must complete this form and attach it to their project proposal document. There are two parts:

PART A: Ethics Checklist. All students must complete this part.

The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

PART B: Ethics Proportionate Review Form. Students who have answered "no" to all questions in A1, A2 and A3 and "yes" to question 4 in A4 in the ethics checklist must complete this part. The project supervisor has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk. The approval may be ***provisional*** – *identifying the planned research as* likely to involve MINIMAL RISK. In such cases you must additionally seek ***full approval*** from the supervisor as the project progresses and details are established. ***Full approval*** must be acquired in writing, before beginning the planned research.

A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
1.1	Does your research require approval from the National Research Ethics Service (NRES)? *e.g. because you are recruiting current NHS patients or staff?* *If you are unsure try - https://www.hra.nhs.uk/approvals-amendments/what-approvals-do-i-need/*	**NO**

1.2	Will you recruit participants who fall under the auspices of the Mental Capacity Act? *Such research needs to be approved by an external ethics committee such as NRES or the Social Care Research Ethics Committee - http://www.scie.org.uk/research/ethics-committee/*	**NO**
1.3	Will you recruit any participants who are currently under the auspices of the Criminal Justice System, for example, but not limited to, people on remand, prisoners and those on probation? *Such research needs to be authorised by the ethics approval system of the National Offender Management Service.*	**NO**
A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
2.1	Does your research involve participants who are unable to give informed consent? *For example, but not limited to, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf.*	**NO**
2.2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	**NO**
2.3	Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)?	**NO**
2.4	Does your project involve participants disclosing information about special category or sensitive subjects? *For example, but not limited to: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and proceedings*	**NO**
2.5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study? *Please check the latest guidance from the FCO - http://www.fco.gov.uk/en/*	**NO**

2.6	Does your research involve invasive or intrusive procedures? *These may include, but are not limited to, electrical stimulation, heat, cold or bruising.*	**NO**
2.7	Does your research involve animals?	**NO**
2.8	Does your research involve the administration of drugs, placebos or other substances to study participants?	**NO**
A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/ **Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.**		*Delete as appropriate*
3.1	Does your research involve participants who are under the age of 18?	**NO**
3.2	Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)? *This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.*	**NO**
3.3	Are participants recruited because they are staff or students of City, University of London? *For example, students studying on a particular course or module.* *If yes, then approval is also required from the Head of Department or Programme Director.*	**NO**
3.4	Does your research involve intentional deception of participants?	**NO**
3.5	Does your research involve participants taking part without their informed consent?	**NO**
3.5	Is the risk posed to participants greater than that in normal working life?	**NO**
3.7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	**NO**
A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK.		*Delete as appropriate*

If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form. **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.**		
4	Does your project involve human participants or their identifiable personal data? *For example, as interviewees, respondents to a survey or participants in testing.*	**Yes**

PART B: Ethics Proportionate Review Form

If you answered YES to question 4 and NO to all other questions in sections A1, A2 and A3 in PART A of this form, then you may use PART B of this form to submit an application for a proportionate ethics review of your project. Your project supervisor has delegated authority to review and approve this application under proportionate review. You must receive final approval from your supervisor in writing before beginning the planned research.

However, if you cannot provide all the required attachments (see B.3) with your project proposal (e.g. because you have not yet written the consent forms, interview schedules etc), the approval from your supervisor will be ***provisional***. You **must** submit the missing items to your supervisor for approval prior to commencing these parts of your project. Once again, you must receive written confirmation from your supervisor that any provisional approval has been superseded by with ***full approval*** of the planned activity as detailed in the full documents. **Failure to follow this procedure and demonstrate that final approval has been achieved may result in you failing the project module.**

Your supervisor may ask you to submit a full ethics application through Research Ethics Online, for instance if they are unable to approve your application, if the level of risks associated with your project change, or if you need an approval letter from the CSREC for an external organisation

B.1 The following questions must be answered fully. **All grey instructions must be removed.**		*Delete as appropriate*
1.1.	Will you ensure that participants taking part in your project are fully informed about the purpose of the research?	**YES**
1.2	Will you ensure that participants taking part in your project are fully informed about the procedures affecting them or affecting any information	**YES**

	collected about them, including information about how the data will be used, to whom it will be disclosed, and how long it will be kept?	
1.3	When people agree to participate in your project, will it be made clear to them that they may withdraw (i.e. not participate) at any time without any penalty?	**YES**
1.4	Will consent be obtained from the participants in your project? Consent from participants will be necessary if you plan to involve them in your project or if you plan to use identifiable personal data from existing records. "Identifiable personal data" means data relating to a living person who might be identifiable if the record includes their name, username, student id, DNA, fingerprint, address, etc. *If YES, you must attach drafts of the participant information sheet(s) and consent form(s) that you will use in section B.3 or, in the case of an existing dataset, provide details of how consent has been obtained.* *You must also retain the completed forms for subsequent inspection. Failure to provide the completed consent request forms will result in withdrawal of any earlier ethical approval of your project.*	**YES**
1.5	Have you made arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential?	**YES**

B.2 If the answer to the following question (B2) is YES, you must provide details		*Delete as appropriate*
2	Will the research be conducted in the participant's home or other non-University location? *If **YES**, you must provide details of how your safety will be ensured.*	**NO**

B.3 Attachments **ALL of the following documents MUST be provided to supervisors if applicable.** **All must be considered prior to final approval by supervisors.**	***YES***	***NO***	***Not Applicable***

A written record of final approval must be provided and retained.			
Details on how safety will be assured in any non-University location, including risk assessment if required (see B2)			**X**
Details of arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential (see B1.5) *Any personal data must be acquired, stored and made accessible* *in ways that are GDPR compliant.*	**X**		
Full protocol for any workshops or interviews**			**X**
Participant information sheet(s)**	**X**		
Consent form(s)**	**X**		
Questionnaire(s)** *sharing a Qualtrics survey with your supervisor is recommended.*			**X**
Topic guide(s) for interviews and focus groups**			**X**
Permission from external organisations or Head of Department** *e.g. for recruitment of participants*			**X**

***If these items are not available at the time of submitting your project proposal, then **provisional approval** can still be given, under the condition that you must submit the final versions of all items to your supervisor for approval at a later date. **All** such items **must** be seen and approved by your supervisor before the activity for which they are needed begins. Written evidence of **final approval** of your planned activity must be acquired from your supervisor before you commence.*

Changes

If your plans change and any aspects of your research that are documented in the approval process change as a consequence, then any approval acquired is invalid. If issues addressed in Part A (the checklist) are affected, then you must complete the approval process again and establish the kind of approval that is required. If issues addressed in Part B are affected, then you must forward updated documentation to your supervisor and have received written confirmation of approval of the revised activity before proceeding.

Templates for Consent and Information

You must use the templates provided by the University as the basis for your participant information sheets and consent forms. You **must** adapt them according to the needs of your project before you submit them for consideration.

Participant Information Sheets, Consent Forms and Protocols must be consistent. Please ensure that this is the case prior to seeking approval. Failure to do so will slow down the approval process.

We strongly recommend using Qualtrics to produce digital information sheets and consent forms.

Appendix B Consent & Participant Information Forms (Collaborating Experts)

PARTICIPANT INFORMATION SHEET

ETH2021-2068, 3/6/2021 v1

Knowledge Graph of Food Policy Networks: Construction, Visualisation & Refinement

Name of principal investigator/researcher: Ollie Keers

We would like to invite you to take part in a research study. Before you decide whether you would like to take part it is important that you understand why the research is being done and what it would involve for you. Please take time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information. You will be given a copy of this information sheet to keep.

What is the purpose of the study?

The study aims to model the complex and interconnected systems within the landscape of food policy, as understood by experts. The study lasts four months, as research for a dissertation that is part of the MSc in Data Science.

Why have I been invited to take part?

You have been invited to take part as an expert on food policy.

Do I have to take part?

Your participation in this project is entirely voluntary, and you can choose not to participate in part or all of the project without being penalised or disadvantaged in any way. It is up to you to decide whether or not to take part. If you do decide to take part you will be asked to sign a consent form. If you decide to take part you are still free to withdraw at any time and without giving a reason. Once your data has been received you will no longer be able to withdraw this data.

What will happen if I take part?

You will be provided with material for review in your own time. This may be videos, images, computer software or other related work.

You will participate in virtual meetings with the researcher, and potentially other collaborators.

You will be asked to provide your opinions, views, reactions and requirements in response to the material you have been provided with.

These may be recorded, quoted anonymously in publication, and used to further develop the software.

These meetings will continue until the conclusion of the study, at the end of September.

What are the possible disadvantages and risks of taking part?

There is a risk that your identity as a participant may be revealed, despite the measures in place to protect privacy which are outlined below. You have been selected for this study via Wren & Co who are collaborating with this study. As such, Wren and Co. will be named in any publications. While all quotes used in publication will not be attributed to you personally, it is possible that you may be identifiable and any quotes attributed to you, correctly or incorrectly by a reader. While very unlikely given the data that will be acquired for this work, this has the potential to harm your reputation.

What are the possible benefits of taking part?

Through participating, your views and requirements will shape the development of this study. This has the potential to make the final product more useful for you. The project has the potential to foster a greater understanding of food policy networks for other audiences, and so provide a wider societal benefit as well.

Expenses and Payments

You will not receive any expenses, payments or rewards for participating in this study,

How is the project being funded?

This project is funded from the Higher Education Innovation Fund (HEIF)

Conflicts of interests

There are no notable conflicts of interest.

What should I do if I want to take part?

Complete and return the attached consent form via email.

Data privacy statement

City, University of London is the sponsor and the data controller of this study based in the United Kingdom. This means that we are responsible for looking after your information and using it properly. The legal basis under which your data will be processed is City's public task.

Your right to access, change or move your information are limited, as we need to manage your information in a specific way in order for the research to be reliable and accurate. To safeguard your rights, we will use the minimum personal-identifiable information possible (for further information please see https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/lawful-basis-for-processing/public-task/).

City will use your name and contact details to contact you about the research study as necessary. If you wish to receive the results of the study, your contact details will also be kept for this purpose. The only people at City who will have access to your identifiable information will be those involved in this project. City will keep identifiable information about you from this study for 1 year after the study has finished.

You can find out more about how City handles data by visiting https://www.city.ac.uk/about/governance/legal. If you are concerned about how we have processed your personal data, you can contact the Information Commissioner's Office (IOC) https://ico.org.uk/.

Will my taking part in the study be kept confidential?

Any quotes will be anonymized before publication.

Prior to anonymization, your data may be accessible to the researcher, supervisors and collaborators.

Records of your data will be stored on a hard drive, and a private GitHub repository for the duration of the study, before being destroyed.

Beyond anonymous quotation, your data will not be shared.

What will happen to the results?

This research is for a Masters dissertation, and may result in further publication in academic journals or presentation at conferences. Your anonymity will be maintained if further publication does occur.

Who has reviewed the study?

This study has been approved by City, University of London Computer Science Research Ethics Committee.

What if there is a problem?

If you have any problems, concerns or questions about this study, you should ask to speak to a member of the research team. If you remain unhappy and wish to complain formally, you can

do this through City's complaints procedure. To complain about the study, you need to phone 020 7040 3040. You can then ask to speak to the Secretary to Senate Research Ethics Committee and inform them that the name of the project is Knowledge Graph of Food Policy Networks: Construction, Visualisation & Refinement

You can also write to the Secretary at:

Anna Ramberg

Research Integrity Manager

City, University of London, Northampton Square

London, EC1V 0HB

Email: Anna.Ramberg.1@city.ac.uk

Insurance

City University London holds insurance policies which apply to this study, subject to the terms and conditions of the policy. If you feel you have been harmed or injured by taking part in this study you may be eligible to claim compensation. This does not affect your legal rights to seek compensation. If you are harmed due to someone's negligence, then you may have grounds for legal action.

Further information and contact details

Researcher: Ollie Keers, ollie.keers.2@city.ac.uk

Supervisor: Dr. Radu Jianu, radu.jianu@city.ac.uk

Thank you for taking the time to read this information sheet.

INFORMED CONSENT FORM

Name of principal investigator/researcher

Ollie Keers

REC reference number

ETH2021-2068

Title of study

Knowledge Graph of Food Policy Networks: Construction, Visualisation & Refinement

Please tick or initial box

1	I confirm that I have read and understood the participant information dated 3/6/2021 v1 for the above study. I have had the opportunity to consider	

	the information and ask questions which have been answered satisfactorily.	
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	
3.	I understand that I will be able to withdraw my data up to the time of submission.	
4.	I agree to the interviews being video recorded.	
5.	I agree to my direct quotes being published anonymously.	
5.	I agree to City recording and processing this information about me. I understand that this information will be used only for the purpose(s) explained in the participant information and my consent is conditional on City complying with its duties and obligations under the General Data Protection Regulation (GDPR).	
6.	I agree to take part in the above study.	

_____ _____ _____

Name of Participant Signature Date

_____ _____ _____

Name of Researcher Signature Date

When completed, 1 copy for participant; 1 copy for researcher file.

Appendix C Consent & Participant Information Forms (Independent Experts)

PARTICIPANT INFORMATION SHEET

ETH2021-2068, 31/8/2021 v2

Knowledge Graph of Food Policy Networks: Construction, Visualisation & Refinement

Name of principal investigator/researcher: Ollie Keers

We would like to invite you to take part in a research study. Before you decide whether you would like to take part it is important that you understand why the research is being done and what it would involve for you. Please take time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information. You will be given a copy of this information sheet to keep.

What is the purpose of the study?

The study aims to model the complex and interconnected systems within the landscape of food policy, as understood by experts. The study lasts four months, as research for a dissertation that is part of the MSc in Data Science.

Why have I been invited to take part?

You have been invited to take part as an expert in your field.

Do I have to take part?

Your participation in this project is entirely voluntary, and you can choose not to participate in part or all of the project without being penalised or disadvantaged in any way. It is up to you to decide whether or not to take part. If you do decide to take part you will be asked to sign a consent form. If you decide to take part you are still free to withdraw at any time and without giving a reason. Once your data has been received you will no longer be able to withdraw this data.

What will happen if I take part?

You will participate in a single, hour-long, virtual meeting with the researcher, and other collaborators.

You will be asked to share your screen, provided with software to use, and given a series of analytical tasks to complete using the software (with support from the researcher).

You will be asked to provide your opinions, views, reactions and requirements in response to the material you have been provided with.

This meeting will be audio recorded, and you may be quoted anonymously in publication.

What are the possible disadvantages and risks of taking part?

There is a risk that your identity as a participant may be revealed, despite the measures in place to protect privacy which are outlined below. You have been selected for this study via Wren & Co who are collaborating with this study. As such, Wren and Co. will be named in any publications. While all quotes used in publication will not be attributed to you personally, it is possible that you may be identifiable and any quotes attributed to you, correctly or incorrectly by a reader. While very unlikely given the data that will be acquired for this work, this has the potential to harm your reputation.

What are the possible benefits of taking part?

Through participating, your views and requirements will shape the development of this study. This has the potential to make the final product more useful for you. The project has the potential to foster a greater understanding of food policy networks for other audiences, and so provide a wider societal benefit as well.

Expenses and Payments

You will not receive any expenses, payments or rewards for participating in this study,

How is the project being funded?

This project is funded from the Higher Education Innovation Fund (HEIF)

Conflicts of interests

There are no notable conflicts of interest.

What should I do if I want to take part?

Complete and return the attached consent form via email.

Data privacy statement

City, University of London is the sponsor and the data controller of this study based in the United Kingdom. This means that we are responsible for looking after your information and using it properly. The legal basis under which your data will be processed is City's public task.

Your right to access, change or move your information are limited, as we need to manage your information in a specific way in order for the research to be reliable and accurate. To safeguard your rights, we will use the minimum personal-identifiable information possible (for further

information please see https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/lawful-basis-for-processing/public-task/).

City will use your name and contact details to contact you about the research study as necessary. If you wish to receive the results of the study, your contact details will also be kept for this purpose. The only people at City who will have access to your identifiable information will be those involved in this project. City will keep identifiable information about you from this study for 1 year after the study has finished.

You can find out more about how City handles data by visiting https://www.city.ac.uk/about/governance/legal. If you are concerned about how we have processed your personal data, you can contact the Information Commissioner's Office (IOC) https://ico.org.uk/.

Will my taking part in the study be kept confidential?

Any quotes will be anonymized before publication.

Prior to anonymization, your data may be accessible to the researcher, supervisors and collaborators.

Records of your data will be stored on a hard drive, and a private GitHub repository for the duration of the study, before being destroyed.

Beyond anonymous quotation, your data will not be shared.

What will happen to the results?

This research is for a Masters dissertation, and may result in further publication in academic journals or presentation at conferences. Your anonymity will be maintained if further publication does occur.

Who has reviewed the study?

This study has been approved by City, University of London Computer Science Research Ethics Committee.

What if there is a problem?

If you have any problems, concerns or questions about this study, you should ask to speak to a member of the research team. If you remain unhappy and wish to complain formally, you can do this through City's complaints procedure. To complain about the study, you need to phone 020 7040 3040. You can then ask to speak to the Secretary to Senate Research Ethics

Committee and inform them that the name of the project is Knowledge Graph of Food Policy Networks: Construction, Visualisation & Refinement

You can also write to the Secretary at:

Anna Ramberg

Research Integrity Manager

City, University of London, Northampton Square

London, EC1V 0HB

Email: Anna.Ramberg.1@city.ac.uk

Insurance

City University London holds insurance policies which apply to this study, subject to the terms and conditions of the policy. If you feel you have been harmed or injured by taking part in this study you may be eligible to claim compensation. This does not affect your legal rights to seek compensation. If you are harmed due to someone's negligence, then you may have grounds for legal action.

Further information and contact details

Researcher: Ollie Keers, ollie.keers.2@city.ac.uk

Supervisor: Dr. Radu Jianu, radu.jianu@city.ac.uk

Thank you for taking the time to read this information sheet.

INFORMED CONSENT FORM

Name of principal investigator/researcher

Ollie Keers

REC reference number

ETH2021-2068

Title of study

Knowledge Graph of Food Policy Networks: Construction, Visualisation & Refinement

Please tick or initial box

1	I confirm that I have read and understood the participant information dated 31/8/2021 v2 for the above study. I have had the opportunity to consider the information and ask questions which have been answered satisfactorily.	

2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	
3.	I understand that I will be able to withdraw my data up to the time of submission.	
4.	I agree to share my screen	
4.	I agree to the interviews being audio recorded.	
5.	I agree to my direct quotes being published anonymously.	
5.	I agree to City recording and processing this information about me. I understand that this information will be used only for the purpose(s) explained in the participant information and my consent is conditional on City complying with its duties and obligations under the General Data Protection Regulation (GDPR).	
6.	I agree to take part in the above study.	

_____	_____	_____
Name of Participant	Signature	Date

_____	_____	_____
Name of Researcher	Signature	Date

When completed, 1 copy for participant; 1 copy for researcher file.

Appendix D Interview Protocol

Entire session recorded on Audacity, audio only. Stored on Researcher's HD, deleted at conclusion of study.

Conducted on zoom with participant to share screen.

Start meeting @ :55, make cohosts,

Brief intro: **~3 mins**:

- Hi, I'm Ollie Keers – MSc Data Science Student at City, virtual internship with Wren & Co
- **About to start audio recording, check that's ok? No zoom notification**
- **I've started audio only recording – let me know if you would like to stop. GI centre will drop off, just you and I, W&co here to assist with anything policy related I can't answer!**
- Objectives for dev: Brief from W&Co: *"informative policy map of UK food policy"*, Researcher: "*visualisation to support the reflective and reasoning process within food policy"*
- note prototype nature of software & currently limited **WASTE** data underpinning vis, from domain researcher extracting from documents.
- outline format for session (~15 mins guided analytical tasks/tutorial, ~5 mins of free use, ~15 mins discussion, <5 mins questionnaire)
- Send link: https://fp-edit.herokuapp.com
- Ask participant to share screen

- Legend at bottom, nodes are blobs, edges are links.
- Ask if there are any unfamiliar terms etc.

Guided Tour/User Tasks (~**12 mins** including researcher giving instructions & support verbally) Green tasks can be dropped if pressed for time, orange tasks can be heavily directed.

Completion (bool) & Time(s) also recorded:

1. Zoom in and identify one actor, one action and one issue contained within the data.
2. For one of these Actors or Actions, click on the node to select it. Click on the 'Selected Node Data' tab. What other classes does it belong to?
3. Choose any two connected nodes. Please create a statement using the label on the arrow between them? ***Can you comment on that statement?***
4. Use the Display Options tab: click 'Show All Data" to include literature review data. Use the checkboxes to show all classes of nodes again. Click an action (lime green) to select it. On the Navigation tab, "Focus on selected Node". Identify one piece of information/statement now available from this view. ***Can you comment on that statement?***
5. Select XXXXX [node to be selected by researcher]. "Expand selected node". ***Take a moment to have a look at everything currently on screen. Do you have any thoughts/questions about what you see?***
6. Search for "food waste crime". Two of the actors are "The Secretary of State for Environment, Food and Rural Affairs" and 'DEFRA". Click one, hold shift and click the other. Use the Editing Toolkit to create any true, labelled, relationship between the two. ***Without adding them, are there any other relationships or nodes that you think should be here?***
7. ***Do you see any problems with the nodes that are currently displayed?*** [Two 'Food Industry" nodes"]. Use the editing toolkit to delete one.
8. Use the "Navigation Tab' to Load Class Schema. Look at the 'Actors'. There is a Government with a capital 'G' and one with a lowercase 'g'. These represent different things. ***What do you think the difference is? What does this view tell you?***
9. On the 'Navigation' tab, show the Actor-Action-Actor paths. From the data here, which Actor has taken the most Actions, and who is impacted by Actions the most? ***Why might this be the case?***
10. On the 'Navigation' tab, show the Actor-Action-Issue paths (takes around 20 seconds to load). Identify an issue (blue) that many actions relate to, and one that very few actions relate to. ***Why do you think this might be****?*

Participant to take **up to 5 minutes** to use the tool freely **to investigate an aspect of food waste of interest to you**, with support offered by both the researcher (and if needed the collaborating domain expert), talking aloud about any thoughts, ideas or questions that come up.

FREE PLAY HARD STOP AT HH:35.

Discussion (**10 mins – 1.5 mins per q**) participant **can continue to use tool** if they wish

- Has this tool confirmed anything you already knew?
- Have you learned anything new or useful from using this tool?
- Has this tool affected how you think about food policy? How so/how did it achieve this?
- What aspects of the tool (graphics/interactions) do you think would be most useful to you? Why?
- Would you use this tool regularly to support your work? How?
- Any other general comments (positive or negative)?

HH:45.

STOP PARTICIPANT SCREEN SHARE

Participant to Complete SUS after interview (in own time if needed) (**~2 mins**):

HARD STOP at H:50

Appendix E Visualisation URL

https://fp-edit.herokuapp.com

N.B. Due to the free hosting, this takes ~1 minute to load if it has not been accessed recently.

Running off code stored at https://github.com/Ollie-K/fp-edit

Appendix F Code, Data & Intermediate Results

4 major pieces of code will be provided here: the final created ontology (before combination with data), the script to generate the knowledge graph, the script to visualise the OWL2Vec* output embeddings, and the visualisation script.

All other code, intermediate results, data etc. are available at: https://github.com/Ollie-K/Domain-Knowledge-Graph-Visualisation

1 - Final Ontology (onto-13-8.owl)

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://wrenand.co.uk/fpn/"
     xml:base="http://wrenand.co.uk/fpn/"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="http://wrenand.co.uk/fpn/"/>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->



    <!-- http://wrenand.co.uk/fpn/actionImpacts -->
```

```
<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/actionImpacts"/>



<!-- http://wrenand.co.uk/fpn/activeIn -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/activeIn">
   <rdfs:domain rdf:resource="http://wrenand.co.uk/fpn/Actor"/>
   <rdfs:range rdf:resource="http://wrenand.co.uk/fpn/Issue"/>
   <owl:propertyChainAxiom rdf:parseType="Collection">
      <rdf:Description>
         <owl:inverseOf rdf:resource="http://wrenand.co.uk/fpn/actionImpacts"/>
      </rdf:Description>
      <rdf:Description rdf:about="http://wrenand.co.uk/fpn/relatesTo"/>
   </owl:propertyChainAxiom>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/advise -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/advise">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/actionImpacts"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/amend -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/amend">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/actionImpacts"/>
</owl:ObjectProperty>
```

```
<!-- http://wrenand.co.uk/fpn/awardedTo -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/awardedTo">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/actionImpacts"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/concernsPlace -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/concernsPlace">
   <rdfs:domain rdf:resource="http://wrenand.co.uk/fpn/Action"/>
   <rdfs:range rdf:resource="http://wrenand.co.uk/fpn/Place"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/deliberateOn -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/deliberateOn">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/takeAction"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/formWorkingGroups -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/formWorkingGroups">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/actionImpacts"/>
</owl:ObjectProperty>
```

```
<!-- http://wrenand.co.uk/fpn/implement -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/implement">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/takeAction"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/interactWith -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/interactWith">
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
   <owl:propertyChainAxiom rdf:parseType="Collection">
      <rdf:Description rdf:about="http://wrenand.co.uk/fpn/takeAction"/>
      <rdf:Description rdf:about="http://wrenand.co.uk/fpn/actionImpacts"/>
   </owl:propertyChainAxiom>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/invite -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/invite">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/takeAction"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/issueIn -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/issueIn">
   <rdfs:domain rdf:resource="http://wrenand.co.uk/fpn/Issue"/>
   <rdfs:range rdf:resource="http://wrenand.co.uk/fpn/Place"/>
```

```
    <owl:propertyChainAxiom rdf:parseType="Collection">
      <rdf:Description>
        <owl:inverseOf rdf:resource="http://wrenand.co.uk/fpn/relatesTo"/>
      </rdf:Description>
      <rdf:Description rdf:about="http://wrenand.co.uk/fpn/concernsPlace"/>
    </owl:propertyChainAxiom>
  </owl:ObjectProperty>



  <!-- http://wrenand.co.uk/fpn/launch -->

  <owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/launch">
    <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/takeAction"/>
  </owl:ObjectProperty>



  <!-- http://wrenand.co.uk/fpn/locatedIn -->

  <owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/locatedIn">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
    <rdfs:range rdf:resource="http://wrenand.co.uk/fpn/Place"/>
  </owl:ObjectProperty>



  <!-- http://wrenand.co.uk/fpn/makePolicyRecommendationsFor -->

  <owl:ObjectProperty
rdf:about="http://wrenand.co.uk/fpn/makePolicyRecommendationsFor">
    <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/actionImpacts"/>
  </owl:ObjectProperty>
```

```
<!-- http://wrenand.co.uk/fpn/makeRecommendationsFor -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/makeRecommendationsFor">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/actionImpacts"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/measure -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/measure">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/takeAction"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/modelAndExtrapolate -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/modelAndExtrapolate">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/actionImpacts"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/provide -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/provide">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/takeAction"/>
</owl:ObjectProperty>
```

```
<!-- http://wrenand.co.uk/fpn/publish -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/publish">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/takeAction"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/publishedIn -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/publishedIn">
   <rdfs:domain rdf:resource="http://wrenand.co.uk/fpn/Action"/>
   <rdfs:range rdf:resource="http://wrenand.co.uk/fpn/Publication"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/raiseAwarenessOf -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/raiseAwarenessOf">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/actionImpacts"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/relatesTo -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/relatesTo">
   <rdfs:domain rdf:resource="http://wrenand.co.uk/fpn/Action"/>
   <rdfs:range rdf:resource="http://wrenand.co.uk/fpn/Issue"/>
</owl:ObjectProperty>
```

```
<!-- http://wrenand.co.uk/fpn/resultIn -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/resultIn">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/takeAction"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/takeAction -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/takeAction"/>



<!-- http://wrenand.co.uk/fpn/usesMethod -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/usesMethod">
   <rdfs:range rdf:resource="http://wrenand.co.uk/fpn/Method"/>
</owl:ObjectProperty>



<!-- http://wrenand.co.uk/fpn/workingOn -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/workingOn">
   <rdfs:domain rdf:resource="http://wrenand.co.uk/fpn/Actor"/>
   <rdfs:range rdf:resource="http://wrenand.co.uk/fpn/Issue"/>
   <owl:propertyChainAxiom rdf:parseType="Collection">
      <rdf:Description rdf:about="http://wrenand.co.uk/fpn/takeAction"/>
      <rdf:Description rdf:about="http://wrenand.co.uk/fpn/relatesTo"/>
   </owl:propertyChainAxiom>
</owl:ObjectProperty>
```

```
<!-- http://wrenand.co.uk/fpn/wrote -->

<owl:ObjectProperty rdf:about="http://wrenand.co.uk/fpn/wrote">
   <rdfs:subPropertyOf rdf:resource="http://wrenand.co.uk/fpn/takeAction"/>
</owl:ObjectProperty>



<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Data properties
//
///////////////////////////////////////////////////////////////////////////////////
 -->



<!-- http://wrenand.co.uk/fpn/citations -->

<owl:DatatypeProperty rdf:about="http://wrenand.co.uk/fpn/citations">
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
</owl:DatatypeProperty>



<!-- http://wrenand.co.uk/fpn/date -->

<owl:DatatypeProperty rdf:about="http://wrenand.co.uk/fpn/date">
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
</owl:DatatypeProperty>
```

```
<!-- http://wrenand.co.uk/fpn/impactFactor -->

<owl:DatatypeProperty rdf:about="http://wrenand.co.uk/fpn/impactFactor">
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#double"/>
</owl:DatatypeProperty>



<!-- http://wrenand.co.uk/fpn/peerReviewed -->

<owl:DatatypeProperty rdf:about="http://wrenand.co.uk/fpn/peerReviewed">
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</owl:DatatypeProperty>



<!-- http://wrenand.co.uk/fpn/significance -->

<owl:DatatypeProperty rdf:about="http://wrenand.co.uk/fpn/significance">
   <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</owl:DatatypeProperty>



<!-- http://wrenand.co.uk/fpn/source -->

<owl:DatatypeProperty rdf:about="http://wrenand.co.uk/fpn/source">
   <rdfs:domain rdf:resource="http://wrenand.co.uk/fpn/Action"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI"/>
</owl:DatatypeProperty>
```

```
<!--
///////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////
 -->



<!-- http://wrenand.co.uk/fpn/Academic -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Academic">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Person"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Action -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Action"/>



<!-- http://wrenand.co.uk/fpn/Activity -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Activity">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Action"/>
</owl:Class>
```

```
<!-- http://wrenand.co.uk/fpn/Actor -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Actor"/>



<!-- http://wrenand.co.uk/fpn/Advice -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Advice">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Action"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Article -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Article">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Action"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Book_Chapter -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Book_Chapter">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Publication"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Business -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Business">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Group"/>
```

```
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Coalition -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Coalition">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Group"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Community -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Community">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Group"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Conference_Paper -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Conference_Paper">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Publication"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Convention -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Convention">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Event"/>
</owl:Class>
```

```
<!-- http://wrenand.co.uk/fpn/Country -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Country">
   <owl:equivalentClass rdf:resource="http://wrenand.co.uk/fpn/Nation"/>
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Group"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Deliberation -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Deliberation">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Policy"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Entrepreneur -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Entrepreneur">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Person"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Event -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Event">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Action"/>
</owl:Class>
```

```
<!-- http://wrenand.co.uk/fpn/Funding -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Funding">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Policy"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Government -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Government">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Group"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Government_Department -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Government_Department">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Government"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Group -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Group">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Actor"/>
   <owl:disjointWith rdf:resource="http://wrenand.co.uk/fpn/Person"/>
</owl:Class>
```

```
<!-- http://wrenand.co.uk/fpn/Guidelines -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Guidelines">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Policy"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Intergovernmental_Organization -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Intergovernmental_Organization">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Group"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Issue -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Issue"/>



<!-- http://wrenand.co.uk/fpn/Journal -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Journal">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Publication"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Magazine_Article -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Magazine_Article">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Publication"/>
```

```
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Method -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Method"/>



<!-- http://wrenand.co.uk/fpn/Nation -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Nation">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Group"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Person -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Person">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Actor"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Place -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Place"/>



<!-- http://wrenand.co.uk/fpn/Policy -->
```

```
<owl:Class rdf:about="http://wrenand.co.uk/fpn/Policy">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Action"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Policy_Paper_or_Report -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Policy_Paper_or_Report">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Publication"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Political_Party -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Political_Party">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Government"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Poster -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Poster">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Publication"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Press_Release -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Press_Release">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Action"/>
```

```
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Public_Benefit -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Public_Benefit">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Group"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Publication -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Publication"/>



<!-- http://wrenand.co.uk/fpn/Question -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Question">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Action"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Recommendation -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Recommendation">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Action"/>
</owl:Class>
```

```
<!-- http://wrenand.co.uk/fpn/Research_Institute -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Research_Institute">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Group"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Scientist -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Scientist">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Person"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Study -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Study">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Recommendation"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Thesis -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/Thesis">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Publication"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/Toolkit -->
```

```
<owl:Class rdf:about="http://wrenand.co.uk/fpn/Toolkit">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Action"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/government -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/government">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Person"/>
</owl:Class>



<!-- http://wrenand.co.uk/fpn/intergovernmental -->

<owl:Class rdf:about="http://wrenand.co.uk/fpn/intergovernmental">
   <rdfs:subClassOf rdf:resource="http://wrenand.co.uk/fpn/Person"/>
</owl:Class>



<!--
///////////////////////////////////////////////////////////////////////////////
//
// General axioms
//
///////////////////////////////////////////////////////////////////////////////
 -->

<rdf:Description>
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AllDisjointClasses"/>
   <owl:members rdf:parseType="Collection">
      <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Action"/>
```

```
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Actor"/>
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Issue"/>
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Method"/>
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Place"/>
        </owl:members>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AllDisjointClasses"/>
        <owl:members rdf:parseType="Collection">
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Action"/>
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Actor"/>
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Issue"/>
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Method"/>
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Place"/>
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Publication"/>
        </owl:members>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AllDisjointClasses"/>
        <owl:members rdf:parseType="Collection">
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Action"/>
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Actor"/>
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Issue"/>
            <rdf:Description rdf:about="http://wrenand.co.uk/fpn/Place"/>
        </owl:members>
    </rdf:Description>
</rdf:RDF>
```

<!-- Generated by the OWL API (version 4.5.9.2019-02-01T07:24:44Z) https://github.com/owlcs/owlapi -->

2 – Knowledge Graph Creation (13-8-ingest.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Aug 13 08:24:19 2021

@author: ollie
"""
from rdflib import Graph
from rdflib import URIRef, Literal
from rdflib import Namespace
import pandas as pd
from datetime import datetime
import re
import owlrl
import json


def createDummyGraph():

    #Empty graph
    g = Graph()
    print('Initialised empty graph')
    #read in the data
    dummy = pd.read_csv('dd-final.csv', header=0)
    dummy = dummy.iloc[1:,:]


    #Creating Namespaces
    fp = Namespace("http://wrenand.co.uk/fpn/")
    rdf = Namespace("http://www.w3.org/1999/02/22-rdf-syntax-ns#")
    rdfs = Namespace("http://www.w3.org/2000/01/rdf-schema#")
    owl = Namespace("http://www.w3.org/2002/07/owl#")
```

```
    #Prefixes
    g.bind("fp", fp)
    g.bind("rdf", rdf)
    g.bind("rdfs", rdfs)
    g.bind("owl", owl)

    for index, row in dummy.iterrows():
        #creating nodes & assigning types based on column locations
          if pd.notnull(row[0]):
                actor_1 = URIRef("http://wrenand.co.uk/fpn/{}".format(row[0].strip().replace(' ',
'_').replace('`', '').replace('\n', '_').replace('\r', '')))
                g.add((actor_1, rdf.type, fp.Actor))
                g.add((actor_1, rdf.type, owl.NamedIndividual))
          if pd.notnull(row[1]):
              actor_1_type = URIRef("http://wrenand.co.uk/fpn/{}".format(row[1].strip().replace('
', '_').replace('`', '').replace('\n', '_')))
              g.add((actor_1_type, rdfs.subClassOf, fp.Actor))
          if pd.notnull(row[2]):
              takeAction                                                                    =
URIRef("http://wrenand.co.uk/fpn/{}".format(row[2].strip().lower().replace(' ', '_').replace('`',
'').replace('\n', '_').replace('ion', 'e').replace('ed', "")))
              g.add((takeAction, rdfs.subPropertyOf, fp.takeAction))
          if pd.notnull(row[3]):
              action = URIRef("http://wrenand.co.uk/fpn/{}".format(row[3].strip().replace('  ',
'_').replace('`', '').replace('\n', '_').replace('\r', '')))
              g.add((action, rdf.type, fp.Action))
          if pd.notnull(row[4]):
              action_type = URIRef("http://wrenand.co.uk/fpn/{}".format(row[4].strip().replace(' ',
'_').replace('`', '').replace('\n', '_')))
              g.add((action_type, rdfs.subClassOf, fp.Action))
          if pd.notnull(row[5]):
              actor_1_loc = URIRef("http://wrenand.co.uk/fpn/{}".format(row[5].strip().replace(' ',
'_').replace('`', '').replace('\n', '_')))
              g.add((actor_1_loc, rdf.type, fp.Place))
```

```
            g.add((actor_1_loc, rdf.type, owl.NamedIndividual))
        if pd.notnull(row[6]):
            #Dealing with inconsistent date formats
            action_date_str = str(row[6])
            action_date_trim = action_date_str.strip().replace(' ', '-').replace('\n', '_').replace('/', '-').replace('January', '01').replace('February', '02').replace('March', '03').replace('April', '04').replace('May', '05').replace('June', '06').replace('July', '07').replace('August', '08').replace('September', '09').replace('October', '10').replace('November', '11').replace('December', '12').replace('Jan', '01').replace('Feb', '02').replace('Mar', '03').replace('Apr', '04').replace('May', '05').replace('Jun', '06').replace('Jul', '07').replace('Aug', '08').replace('Sep', '09').replace('Oct', '10').replace('Nov', '11').replace('Dec', '12')
            if len(action_date_trim) == 4:
                action_date_trim = "01-01-" + action_date_trim
                action_date = datetime.strptime(action_date_trim, '%d-%m-%Y')
            if len(action_date_trim) == 10:
                action_date = datetime.strptime(action_date_trim, '%d-%m-%Y')
            if len(action_date_trim) == 9:
                action_day = action_date_trim.split('-')[0]
                action_month = action_date_trim.split('-')[1]
                action_year = action_date_trim.split('-')[2]
                if len(action_day) == 1:
                    action_day = '0' + action_day
                if len(action_month) == 1:
                    action_month = '0' + action_month
                action_date_trim = action_day + '-' + action_month + '-'+ action_year
                action_date = datetime.strptime(action_date_trim, '%d-%m-%Y')
            if len(action_date_trim) == 8:
                action_date = datetime.strptime(action_date_trim, '%d-%m-%y')
        if pd.notnull(row[7]):
            actionImpacts = URIRef("http://wrenand.co.uk/fpn/{}".format(row[7].strip().lower().replace(' ', '_').replace(`', '').replace('\n', '_')))
            g.add((actionImpacts, rdfs.subPropertyOf, fp.actionImpacts))
        if pd.notnull(row[8]):
```

```
            actor_2 = URIRef("http://wrenand.co.uk/fpn/{}".format(row[8].strip().replace(' ',
'_').replace('`', '').replace('\n', '_')))
            g.add((actor_2, rdf.type, fp.Actor))
            g.add((actor_2, rdf.type, owl.NamedIndividual))
        if pd.notnull(row[9]):
            actor_2_type = URIRef("http://wrenand.co.uk/fpn/{}".format(row[9].strip().replace('
', '_').replace('`', '').replace('\n', '_')))
            g.add((actor_2_type, rdfs.subClassOf, fp.Actor))
        if pd.notnull(row[10]):
            issue                                                                    =
URIRef("http://wrenand.co.uk/fpn/{}".format(row[10].strip().lower().replace('                ',
'_').replace('`', '').replace('\n', '_')))
            g.add((issue, rdf.type, fp.Issue))
            g.add((issue, rdf.type, owl.NamedIndividual))
        if pd.notnull(row[11]):
            descriptor = str(row[11]).strip().replace('\n', ' ').replace('\r', '')
        if pd.notnull(row[12]):
            reference = URIRef(row[12].replace(' ', '').strip().replace('\n', '').replace('\r', ''))

        #creating edges

        if pd.notnull(row[0]) and pd.notnull(row[1]):
            g.add((actor_1, rdf.type, actor_1_type))

        if pd.notnull(row[0]) and pd.notnull(row[2]) and pd.notnull(row[3]):
            g.add((actor_1, takeAction, action))

        if pd.notnull(row[3]) and pd.notnull(row[4]):
            g.add((action, rdf.type, action_type))

        if pd.notnull(row[0]) and pd.notnull(row[5]):
            g.add((actor_1, fp.locatedIn, actor_1_loc))

        if pd.notnull(row[3]) and pd.notnull(row[6]):
```

```
            g.add((action, fp.date, Literal(action_date)))

        if pd.notnull(row[3]) and pd.notnull(row[7]) and pd.notnull(row[8]):
            g.add((action, actionImpacts, actor_2))

        if pd.notnull(row[8]) and pd.notnull(row[9]):
            g.add((actor_2, rdf.type, actor_2_type))

        if pd.notnull(row[3]) and pd.notnull(row[10]):
            g.add((action, fp.relatesTo, issue))

        if pd.notnull(row[0]) and pd.notnull(row[10]):
            g.add((actor_1, fp.workingOn, issue))

        if pd.notnull(row[8]) and pd.notnull(row[10]):
            g.add((actor_2, fp.activeIn, issue))

        if pd.notnull(row[3]) and pd.notnull(row[10]):
            g.add((action, fp.relatesTo, issue))

        if pd.notnull(row[3]) and pd.notnull(row[11]):
            g.add((action, fp.significance, Literal(descriptor)))

        if pd.notnull(row[3]) and pd.notnull(row[12]):
            g.add((action, fp.source, reference))
    print("Created '" + str(len(g)) + "' triples.")
    #saving knowledge graph
    g.serialize(destination='13-8.ttl', format='ttl')


def createDummyLitGraph():

    #Empty graph
    g = Graph()
```

```
    #read in dummy data graph
    g.parse("13-8.ttl", format="ttl")
    print("Loaded '" + str(len(g)) + "' triples.")
    #read in the  csv
    litrev = pd.read_csv('Lit Review Data.csv', header=1)



    #Creating Namespaces
    fp = Namespace("http://wrenand.co.uk/fpn/")
    rdf = Namespace("http://www.w3.org/1999/02/22-rdf-syntax-ns#")
    rdfs = Namespace("http://www.w3.org/2000/01/rdf-schema#")
    owl = Namespace("http://www.w3.org/2002/07/owl#")

    #Prefixes
    g.bind("fp", fp)
    g.bind("rdf", rdf)
    g.bind("rdfs", rdfs)
    g.bind("owl", owl)

    for index, row in litrev.iterrows():
        #creating nodes using columnar matching as before
          if pd.notnull(row[3]):
              article    =    URIRef("http://wrenand.co.uk/fpn/{}".format(row[3].strip().replace('    ',
'_').replace('`', '').replace('\n', '_').replace('\r', '')))
              g.add((article, rdf.type, fp.Article))
              g.add((article, rdf.type, owl.NamedIndividual))
          if pd.notnull(row[4]): #dealing with inconsistent author formats
              if ';' in row[4]:
                  for author in row[4].replace(' and ', '; ').split('; '):
                      author = re.sub(r'[0-9]+', '', author)
                      author_node                                                                =
URIRef("http://wrenand.co.uk/fpn/{}".format(author.strip().replace('        ',        '_').replace('`',
'').replace('\n', '_').replace('\r', '')))
                      g.add((author_node, rdf.type, fp.Academic))
```

```
                g.add((author_node, rdf.type, owl.NamedIndividual))
            elif '.,' in row[4]:
                for author in row[4].replace(' and ', '., ').split("., "):
                    author = re.sub(r'[0-9]+', '', author) + '.'
                    author_node = URIRef("http://wrenand.co.uk/fpn/{}".format(author.strip().replace(' ', '_').replace('`', '').replace('\n', '_').replace('\r', '')))
                    g.add((author_node, rdf.type, fp.Academic))
                    g.add((author_node, rdf.type, owl.NamedIndividual))
            else:
                for author in row[4].replace(' and ', ', ').split(", "):
                    author = author.replace(',', '')
                    author = re.sub(r'[0-9]+', '', author).lstrip()
                    author_node = URIRef("http://wrenand.co.uk/fpn/{}".format(author.strip().replace(' ', '_').replace('`', '').replace('\n', '_').replace('\r', '')))
                    g.add((author_node, rdf.type, fp.Academic))
                    g.add((author_node, rdf.type, owl.NamedIndividual))
        if pd.notnull(row[6]):
            pub_date = datetime.strptime(str(row[6]), '%Y')
        if pd.notnull(row[7]):
            publication = URIRef("http://wrenand.co.uk/fpn/{}".format(row[7].strip().replace(' ', '_').replace('`', '').replace('\n', '_').replace('\r', '')))
            g.add((publication, rdf.type, fp.Publication))
            g.add((publication, rdf.type, owl.NamedIndividual))
        if pd.notnull(row[8]):
            impact_factor = float(row[8])
        if pd.notnull(row[10]):
            citations = int(row[10])
        if pd.notnull(row[11]):
            pub_type = URIRef("http://wrenand.co.uk/fpn/{}".format(row[11].strip().replace(' ', '_').replace('`', '').replace('\n', '_').replace('\r', '')))
            g.add((pub_type, rdfs.subClassOf, fp.Publication))
        if pd.notnull(row[13]):
```

```
            if row[13].lower() == 'yes':
                peer_rev = True
            else:
                peer_rev = False
        if pd.notnull(row[18]):
            method1 = URIRef("http://wrenand.co.uk/fpn/{}".format(row[18].strip().replace(' ',
'_').replace('`', '').replace('\n', '_').replace('\r', '')))
            g.add((method1, rdf.type, fp.Method))
        if pd.notnull(row[19]):
            method2 = URIRef("http://wrenand.co.uk/fpn/{}".format(row[19].strip().replace(' ',
'_').replace('`', '').replace('\n', '_').replace('\r', '')))
            g.add((method2, rdf.type, fp.Method))
        if pd.notnull(row[20]):
            method3 = URIRef("http://wrenand.co.uk/fpn/{}".format(row[20].strip().replace(' ',
'_').replace('`', '').replace('\n', '_').replace('\r', '')))
            g.add((method3, rdf.type, fp.Method))
        if pd.notnull(row[21]):
            method4 = URIRef("http://wrenand.co.uk/fpn/{}".format(row[21].strip().replace(' ',
'_').replace('`', '').replace('\n', '_').replace('\r', '')))
            g.add((method4, rdf.type, fp.Method))
        if pd.notnull(row[22]):
            method5 = URIRef("http://wrenand.co.uk/fpn/{}".format(row[22].strip().replace(' ',
'_').replace('`', '').replace('\n', '_').replace('\r', '')))
            g.add((method5, rdf.type, fp.Method))
        if (pd.notnull(row[23]) and len(row[23]) > 1):
            for place in row[23].split(","):
                place_node =
URIRef("http://wrenand.co.uk/fpn/{}".format(row[23].strip().replace(' ', '_').replace('`',
'').replace('\n', '_').replace('\r', '')))
                g.add((place_node, rdf.type, fp.Place))
                g.add((place_node, rdf.type, owl.NamedIndividual))
        if (pd.notnull(row[24]) and len(row[24]) > 1):
            if row[24] == 'Education':
```

```
            issue = URIRef("http://wrenand.co.uk/fpn/{}_on_food_waste".format(row[24].strip().lower().replace(' ', '_').replace('`', '').replace('\n', '_').replace('\r', '')))
        else:
            issue = URIRef("http://wrenand.co.uk/fpn/{}_food_waste".format(row[24].strip().lower().replace(' ', '_').replace('`', '').replace('\n', '_').replace('\r', '')))
        g.add((issue, rdf.type, fp.Issue))
        g.add((issue, rdf.type, owl.NamedIndividual))

    #creating edges
    if pd.notnull(row[3]) and pd.notnull(row[4]):
        if ';' in row[4]:
            for author in row[4].replace(' and ', '; ').split("; "):
                author = re.sub(r'[0-9]+', '', author)
                author_node = URIRef("http://wrenand.co.uk/fpn/{}".format(author.strip().replace(' ', '_').replace('`', '').replace('\n', '_').replace('\r', '')))
                g.add((author_node, fp.wrote, article))
        elif '.,' in row[4]:
            for author in row[4].replace(' and ', '., ').split("., "):
                author = re.sub(r'[0-9]+', '', author) + '.'
                author_node = URIRef("http://wrenand.co.uk/fpn/{}".format(author.strip().replace(' ', '_').replace('`', '').replace('\n', '_').replace('\r', '')))
                g.add((author_node, rdf.type, fp.Academic))
                g.add((author_node, rdf.type, owl.NamedIndividual))
        else:
            for author in row[4].replace(' and ', ', ').split(", "):
                author = author.replace(',', '')
                author = re.sub(r'[0-9]+', '', author).lstrip()
                author_node = URIRef("http://wrenand.co.uk/fpn/{}".format(author.strip().replace(' ', '_').replace('`', '').replace('\n', '_').replace('\r', '')))
```

```
            g.add((author_node, fp.wrote, article))

    if pd.notnull(row[3]) and pd.notnull(row[6]):
        g.add((article, fp.date, Literal(pub_date)))

    if pd.notnull(row[3]) and pd.notnull(row[7]):
        g.add((article, fp.publishedIn, publication))

    if pd.notnull(row[7]) and pd.notnull(row[8]):
        g.add((publication, fp.impactFactor, Literal(impact_factor)))

    if pd.notnull(row[3]) and pd.notnull(row[10]):
        g.add((article, fp.citations, Literal(citations)))

    if pd.notnull(row[7]) and pd.notnull(row[11]):
        g.add((publication, rdf.type, pub_type))

    if pd.notnull(row[3]) and pd.notnull(row[13]):
        g.add((article, fp.peerReviewed, Literal(peer_rev)))

    if pd.notnull(row[3]) and pd.notnull(row[18]):
        g.add((article, fp.usesMethod, method1))

    if pd.notnull(row[3]) and pd.notnull(row[19]):
        g.add((article, fp.usesMethod, method2))

    if pd.notnull(row[3]) and pd.notnull(row[20]):
        g.add((article, fp.usesMethod, method3))

    if pd.notnull(row[3]) and pd.notnull(row[21]):
        g.add((article, fp.usesMethod, method4))

    if pd.notnull(row[3]) and pd.notnull(row[22]):
        g.add((article, fp.usesMethod, method5))
```

```
        if pd.notnull(row[3]) and pd.notnull(row[23]) and len(row[23]) > 1:
            for place in row[23].split(","):
                place_node = URIRef("http://wrenand.co.uk/fpn/{}".format(row[23].strip().replace(' ', '_').replace(`', '').replace('\n', '_').replace('\r', '')))
                g.add((article, fp.concernsPlace, place_node))

        if pd.notnull(row[3]) and pd.notnull(row[24])and len(row[24]) > 1:
            g.add((article, fp.relatesTo, issue))
    print("Graph now consists of '" + str(len(g)) + "' triples.")
    g.serialize(destination='13-8-lit.ttl', format='ttl')
    #saving expanded knowledge graph


def OWLRLInference():

    g = Graph()
    #import current triples
    g.parse("13-8-lit.ttl", format="ttl")

    print("Loaded '" + str(len(g)) + "' triples.")
    #import ontology
    g.load("onto-13-8.owl",  format="xml")



    #Performs RDFS reasoning
    owlrl.DeductiveClosure(owlrl.OWLRL_Semantics, axiomatic_triples=False, datatype_axioms=False).expand(g)


    print("After inference rules: '" + str(len(g)) + "' triples.")
```

```
    print("\nSaving extended graph")
    #exporting to rdf as this can be read by protege but ttl cannot (needed to check for unsatisfiabilities)
    g.serialize(destination='13-8-inference.rdf', format='xml')
    g.serialize(destination='13-8-inference.ttl', format='ttl')

def elementExtraction():
    g = Graph()
    g.parse("13-8-inference.ttl", format="ttl")
    #loading knowledge graph
    print("Extracting Data")



    print('Extracting Elements')
    #SPARQL query to extract all triples
    qres = g.query(
        """
        SELECT *
        WHERE{
        ?s ?p ?o .

        }
        """ )
    df = pd.DataFrame(columns=['source', 'interaction', 'target'])
    #serializing output into dataframe
    for row in qres:
        df = df.append(
            {'source': row.s, 'interaction': row.p, 'target': row.o}, ignore_index=True)

    print(df.shape)
    #drop repeated triples
    df_out = df.drop_duplicates()
    print(df_out.shape)
```

```
#drop axiomatic & class membership triples
df_out = df_out[~(df_out['interaction'].str.contains('rdf'))]
print(df_out.shape)
#drop other axiomatic & class membership triples
df_out = df_out[~(df_out['interaction'].str.contains('owl'))]
print(df_out.shape)
#remove namespace to improve human readability
df_out = df_out.replace('http://wrenand.co.uk/fpn/', '', regex=True)
print(df_out.shape)
#remove any further duplicated triples, remove any blanks
df_out = df_out.drop_duplicates()
df_out.replace("", float("NaN"), inplace=True)
df_out.dropna(how='any', inplace=True)
data = df_out
#making a single column of all entities
node_table = pd.concat([data['source'], data['target']], axis=0)
print(node_table.shape)
#removing any duplicated nodes
node_table = node_table.drop_duplicates()
print(node_table.shape)


node_types = pd.DataFrame(columns=['node', 'class'])
#running a new SPARQL query for each node to extract a list of its types
for node in node_table:
    node = str(node).replace(' ', '_')
    qres = g.query(
        """
        SELECT  ?node_class
        WHERE{
      <http://wrenand.co.uk/fpn/%s> rdf:type ?node_class .
    }
        """ % node)
    type_list = list()
```

```
        for row in qres:
            #ignore axiomatic types
            if 'owl' not in str(row.node_class):
                type_list.append(row.node_class.replace('http://wrenand.co.uk/fpn/', ''))
        #if no type information has been retrieved for a node, it is a Literal
        if len(type_list) == 0:
             type_list.append('Literal')
        node_types = node_types.append({'node' : node, 'class' : type_list}, ignore_index=True)
    #produce a dictionary with nodes as keys and the type list as values
    type_dict = {}
    for index,row in node_types.iterrows():
        type_dict[row[0]] = row[1]

    #Produce node data in visualisation-friendly format to dump into a json
    nodes = list()
    for node in node_table:
        nodes.append({'data': {'id': str(node), 'label': str(node).replace('_', ' ')}, 'classes':
type_dict[str(node).replace(' ', '_')]})

    #and do the same for edge data
    edges = [
        {'data': {'source': row[0], 'target': row[2], 'label': row[1]}}
        for index, row in data.iterrows()
    ]
    #combine the two, and save as a json
    default_elements = nodes + edges
    with open('13-8_elements_file.json', 'w', encoding ='utf8') as json_file:
        json.dump(default_elements, json_file)
    #Now extracting a new table of all nodes (as before) to use for lexical-similarity search
    print('Extracting Elements')
    qres = g.query(
            """
            SELECT ?s ?o
```

```
        WHERE{
        ?s ?p ?o .

        }
        """ )
    df = pd.DataFrame(columns=['source', 'target'])
    for row in qres:
        df = df.append(
            {'source': row.s, 'target': row.o}, ignore_index=True)

    print(df.shape)
    df_out = df.drop_duplicates()
    print(df_out.shape)
    df_out = df_out.replace('http://wrenand.co.uk/fpn/', '', regex=True)
    print(df_out.shape)
    df_out = df_out.drop_duplicates()
    df_out.replace("", float("NaN"), inplace=True)
    df_out.dropna(how='any', inplace=True)
    data = df_out

    node_table = pd.concat([data['source'], data['target']], axis=0)
    print(node_table.head())
    node_table = node_table.drop_duplicates()
    #create a table for dissimilarity scores, initially all zero, and save as csv
    node_lex = pd.DataFrame(columns=['node', 'score'])
    node_lex['node'] = node_table
    node_lex['score'] = 0
    node_lex.to_csv('13-8-nodes.csv', index=False)
    print(node_lex.shape)

def extractOnto():
    g = Graph()
    print("Extracting Data")
```

```
g.parse("13-8-inference.ttl", format="ttl")



#SPARQL query to extract all classes
print('Extracting Elements')
qres = g.query(
      """
      SELECT *
      WHERE{
      ?s a owl:Class .
      ?s ?p ?o .
      ?o a owl:Class .

      }
      """ )
df = pd.DataFrame(columns=['source', 'interaction', 'target'])
for row in qres:
      df = df.append(
         {'source': row.s, 'interaction': row.p, 'target': row.o}, ignore_index=True)

print(df.shape)
df_out = df.drop_duplicates()
print(df_out.shape)
#removing all namespaces (there are more here to deal with)
df_out = df_out.replace('http://wrenand.co.uk/fpn/', '', regex=True)
df_out = df_out.replace('http://www.w3.org/2002/07/owl#', '', regex=True)
df_out = df_out.replace('http://www.w3.org/1999/02/22-rdf-syntax-ns#', '', regex=True)
df_out = df_out.replace('http://www.w3.org/2000/01/rdf-schema#', '', regex=True)
df_out = df_out.replace('http://www.w3.org/2001/XMLSchema#', '', regex=True)
print(df_out.shape)
#remove reflexive class triples as these are unhelpful for humans
df_out = df_out[(df_out['source'] != df_out['target'])]
#remove disjointedness triples: counterintuitive to visualise
df_out = df_out[(df_out['interaction'] != 'disjointWith')]
```

```
print(df_out.shape)
print(df_out)
df_out = df_out.drop_duplicates()
df_out.replace("", float("NaN"), inplace=True)
df_out.dropna(how='any', inplace=True)
data = df_out

node_table = pd.concat([data['source'], data['target']], axis=0)
print(node_table.shape)
node_table = node_table.drop_duplicates()
print(node_table.shape)

#as before, retrieve all classes for these nodes and serialise into a json
node_types = pd.DataFrame(columns=['node', 'class'])
for node in node_table:
    node = str(node).replace(' ', '_')
    qres = g.query(
        """
        SELECT  ?node_class
        WHERE{
      <http://wrenand.co.uk/fpn/%s> rdfs:subClassOf ?node_class .
    }
        """ % node)
    type_list = list()

    for row in qres:
        if 'owl' not in str(row.node_class):
            type_list.append(row.node_class.replace('http://wrenand.co.uk/fpn/', ''))
    if len(type_list) == 0:
        type_list.append('Literal')
    node_types = node_types.append({'node' : node, 'class' : type_list}, ignore_index=True)
type_dict = {}
for index,row in node_types.iterrows():
    type_dict[row[0]] = row[1]
```

```
    nodes = list()
    for node in node_table:
        nodes.append({'data': {'id': str(node), 'label': str(node).replace('_', ' ')}, 'classes':
type_dict[str(node).replace(' ', '_')]})



    edges = [
        {'data': {'source': row[0], 'target': row[2], 'label': row[1]}}
        for index, row in data.iterrows()
    ]
    default_elements = nodes + edges
    with open('onto_elements.json', 'w', encoding ='utf8') as json_file:
        json.dump(default_elements, json_file)

def extractMap():
    g = Graph()
    g.parse("13-8-inference.ttl", format="ttl")

    print("Extracting Data")

    #Query to extract all classes as before
    print('Extracting Elements')
    qres = g.query(
          """
          SELECT *
          WHERE{
         ?s a owl:Class .
         ?s ?p ?o .
         ?o a owl:Class .

          }
          """ )
```

```
df = pd.DataFrame(columns=['source', 'interaction', 'target'])
for row in qres:
        df = df.append(
            {'source': row.s, 'interaction': row.p, 'target': row.o}, ignore_index=True)
print(df.shape)
#query to extract all object properties that have defined domain and range
metaqres = g.query(
    """
        SELECT *
        WHERE{
       ?p a owl:ObjectProperty .
       ?p rdfs:domain ?dom .
       ?p rdfs:range ?ran .


        }
        """ )
for row in metaqres:
        df = df.append(
            {'source': row.dom, 'interaction': row.p, 'target': row.ran}, ignore_index=True)
#creating triples to enable visualisation of the top-level interactions.
#These have not been extracted becasue they do not have a defined domain and/or range.
df = df.append({'source': 'Actor', 'interaction': 'takeAction', 'target': 'Action'}, ignore_index=True)
df = df.append({'source': 'Action', 'interaction': 'actionImpacts', 'target': 'Actor'}, ignore_index=True)
df = df.append({'source': 'Actor', 'interaction': 'locatedIn', 'target': 'Place'}, ignore_index=True)
df = df.append({'source': 'Article', 'interaction': 'usesMethod', 'target': 'Method'}, ignore_index=True)
df = df.append({'source': 'Article', 'interaction': 'citations', 'target': 'Citations'}, ignore_index=True)
df = df.append({'source': 'Action', 'interaction': 'significance', 'target': 'Description'}, ignore_index=True)
df = df.append({'source': 'Action', 'interaction': 'source', 'target': 'URL'}, ignore_index=True)
```

```
    df = df.append({'source': 'Journal', 'interaction': 'impactFactor', 'target': 'impactFactor'}, ignore_index=True)
    df = df.append({'source': 'Journal', 'interaction': 'peerReviewed', 'target': 'Y/N'}, ignore_index=True)
    df = df.append({'source': 'Action', 'interaction': 'date', 'target': 'Date'}, ignore_index=True)


    #as before, creating a node table, using this to extract classes, and dumping into a json.
    print(df.shape)
    df_out = df.drop_duplicates()
    print(df_out.shape)
    df_out = df_out.replace('http://wrenand.co.uk/fpn/', '', regex=True)
    df_out = df_out.replace('http://www.w3.org/2002/07/owl#', '', regex=True)
    df_out = df_out.replace('http://www.w3.org/1999/02/22-rdf-syntax-ns#', '', regex=True)
    df_out = df_out.replace('http://www.w3.org/2000/01/rdf-schema#', '', regex=True)
    df_out = df_out.replace('http://www.w3.org/2001/XMLSchema#', '', regex=True)
    print(df_out.shape)
    df_out = df_out[(df_out['source'] != df_out['target'])]
    df_out = df_out.append({'source': 'Actor', 'interaction': 'interactWith', 'target': 'Actor'}, ignore_index=True)
    df_out = df_out[(df_out['interaction'] != 'disjointWith')]
    print(df_out.shape)
    print(df_out)
    df_out = df_out.drop_duplicates()
    df_out.replace("", float("NaN"), inplace=True)
    df_out.dropna(how='any', inplace=True)
    data = df_out

    node_table = pd.concat([data['source'], data['target']], axis=0)
    print(node_table.shape)
    node_table = node_table.drop_duplicates()
    print(node_table.shape)
```

```
    node_types = pd.DataFrame(columns=['node', 'class'])
    for node in node_table:
        node = str(node).replace(' ', '_')
        qres = g.query(
            """
            SELECT  ?node_class
            WHERE{
          <http://wrenand.co.uk/fpn/%s> rdfs:subClassOf ?node_class .
        }
            """ % node)
        type_list = list()

        for row in qres:
            if 'owl' not in str(row.node_class):
                type_list.append(row.node_class.replace('http://wrenand.co.uk/fpn/', ''))
        if len(type_list) == 0:
             type_list.append('Literal')
        node_types = node_types.append({'node' : node, 'class' : type_list}, ignore_index=True)
    type_dict = {}
    for index,row in node_types.iterrows():
        type_dict[row[0]] = row[1]


    nodes = list()
    for node in node_table:
        nodes.append({'data': {'id': str(node), 'label': str(node).replace('_', ' ')}, 'classes':
type_dict[str(node).replace(' ', '_')]})


    edges = [
        {'data': {'source': row[0], 'target': row[2], 'label': row[1]}}
        for index, row in data.iterrows()
    ]
    default_elements = nodes + edges
```

```
    with open('map_elements.json', 'w', encoding ='utf8') as json_file:
        json.dump(default_elements, json_file)

createDummyGraph()
createDummyLitGraph()
OWLRLInference()
elementExtraction()
extractOnto()
extractMap()
```

3 – Plotting OWL2Vec* output embeddings (sparql_plot.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Aug 31 10:51:49 2021

@author: ollie
"""
# Load back with memory-mapping = read-only, shared across processes.
from gensim.models import KeyedVectors
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()  # for plot styling
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from rdflib import Graph

g = Graph()
g.parse("13-8-inference.ttl", format="ttl")

wv = KeyedVectors.load("fp_3.embeddings", mmap='r') #loading vectors

vec_dict = {} #initialising an empty dictionary
for key in wv.wv.vocab:  # loop through vocabulary
   vec_dict[key] = wv[key] #fill add the corresponding vector to the dictionary of vocab keys

vec_df = pd.DataFrame(vec_dict) #transform dictionary into dataframe
vec_df = vec_df.transpose() #use vocab as index rather than columns
vec_df['uris'] = vec_df.index.to_series()
vec_uris=vec_df[vec_df['uris'].str.contains('wrenand')]
print(vec_uris.head()) #preview of dataframe

#retrieving node classes
```

```
vec_types = pd.DataFrame(columns=['node', 'class'])
for uri in vec_uris['uris']:
        uri = str(uri).replace(' ', '_')
        qres = g.query(
            """
            SELECT  ?node_class
            WHERE{
           <%s> rdf:type ?node_class .
            FILTER ( ?node_class = fp:Actor || ?node_class = fp:Action|| ?node_class = fp:Issue||
?node_class = fp:Method|| ?node_class = fp:Place|| ?node_class = fp:Publication)
        }
            """ % uri)
        for row in qres:
            if 'owl' not in str(row.node_class):
                superclass = row.node_class.replace('http://wrenand.co.uk/fpn/', '')
        if len(qres) == 0:
             superclass = 'Literal/Property'
        vec_types = vec_types.append({'node' : uri, 'class' : superclass}, ignore_index=True)
type_dict = {}
for index,row in vec_types.iterrows():
        type_dict[row[0]] = row[1]
vec_uris['classes'] = vec_uris['uris'].map(type_dict)
print(vec_uris.iloc[:,:-2].head())
x = vec_uris.iloc[:,:-2]
pca = PCA(n_components=2) #2 components for pca
principalComponents = pca.fit_transform(x) #transforming into 2 components
vec_uris[['principal component 1', 'principal component 2']] = principalComponents
              #creating dataframe
colours = {'Actor':'#fb8072', 'Action': '#b3de69', 'Issue':'#0909ed', 'Method':'#ffff29',
'Place':'#fdb462', 'Publication':'#8dd3c7'}
vec_uris['colour'] =vec_uris['classes'].map(colours)
print(vec_uris.head())
fig, ax = plt.subplots(figsize=(7,7))
ax.set_facecolor('white')
```

```
ax.grid(b=None)
for i in colours.keys(): #for each of the 6 clusters
    sub = vec_uris[vec_uris['classes'] == i] #select a subset of the data with that cluster label
    plt.scatter(x=sub['principal component 1'], y=sub['principal component 2'],
            c=sub['colour'], s=10, marker='o', label = '%s'% i, edgecolors='black', linewidth=0.5,
alpha = 0.8) #plotting labelled data
ax.legend(markerscale=2, ncol=2, loc='upper left', facecolor='white', framealpha=0.5,
fontsize='medium')
plt.show() #showing it
```

4 – Visualisation (edit.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Sep  5 14:42:41 2021

@author: ollie
"""



from rdflib import Graph
import pandas as pd



import os
import json
import numpy as np
import dash
from dash.dependencies import Input, Output, State
import dash_core_components as dcc
import dash_html_components as html

import Levenshtein as lev

import dash_cytoscape as cyto

g = Graph()
#import knowledge graph
g.parse("13-8-inference.ttl", format="ttl")
#import table of all nodes for similarity-scoring
node_lex = pd.read_csv('13-8-nodes.csv')

def sim_score(a, b):
    #function for computing standardised lexical similarity scores
```

```
    #put both terms into lowercase to remove case sensitivity
    a = a.lower()
    b = b.lower()
    #set the denominator as the longer term length to avoid values greater than 1
    denom = len(b)
    if len(a) > len(b):
        denom = len(a)
    #calculate the levenshtein distance as a proportion of the longer term length
    c = (lev.distance(a, b) / denom)
    return c



#setting up dash cytoscape
asset_path = os.path.join(
    os.path.dirname(os.path.abspath(__file__)),
    '..', 'assets'
)

app = dash.Dash(__name__, assets_folder=asset_path)
server = app.server
cyto.load_extra_layouts()

#import json of precomputed positions for class hierarchy view
with open('onto-positions.json') as f:
    onto_elements = json.loads(f.read())

#import json of precomputed positions for class schema view
with open('map_pos.json') as f:
    map_elements = json.loads(f.read())

#default html styling elements
styles = {
    'json-output': {
```

```
        'overflow-y': 'scroll',
        'height': 'calc(0.45*(20vh)',
        'border': 'thin lightgrey solid',
    },
    'tab': {'height': 'calc(20vh-100px)'}
}
#stylesheet for hiding some classes exclusive to Literature Review data
dd_style=[
                {
            'selector': '.Academic',
            'style': {
              'display': 'none'
            }
        },

                  {
            'selector': '.Article',
            'style': {
              'display': 'none'
            }
        },

            {
                'selector': '.Method',
                'style':{
                    'display': 'none',

                    }
                },

        {
            'selector': '.Publication',
            'style': {
                'display': 'none',
```

```
        }
    },
    ]

#default stylesheet
all_style = [
            {
            'selector': 'edge',
            'style': {'width':'1',
                    'content': 'data(label)',
                    'text-rotation':'autorotate',
                    'text-opacity': 0.8,
                    'text-background-shape':'rounded-rectangle',
                    'text-background-color':'white',
                    'text-background-opacity':0.7,
                    'font-family':'helvetica',
                    'font-style':'italic',
                    'font-size': 11,
                    'font-weight':'thin',
                    'target-distance-from-node':'5px',
                    'target-endpoint':'outside-to-node-or-label',
                    'line-opacity': '0.7',
                    'curve-style': 'bezier',
                    'control-point-step-size':'100',
                    'control-point-weight':0.2,
                    'target-arrow-shape': 'vee',
                    'arrow-scale': 2,
                    'min-zoomed-font-size':16,
                    }
            },

            {
            'selector': 'edge[label="equivalentClass"]',
```

```
'style': {
        'visibility': 'hidden'
        }
},

{
'selector': 'node[label="'Thing"']',
'style': {
        'display': 'none'
        }
},

{
'selector': 'node[label="'Nothing"']',
'style': {
        'display': 'none'
        }
},

{
    'selector': '.Actor',
    'style': {
            'width':'20',
            'height':'20',
            'background-color': '#fb8072',
            'border-color':'black',
            'border-width':'1',
            'shape':'star',
            'content': 'data(label)',
            'text-wrap': 'wrap',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
```

```
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
},

 {
    'selector': '.Action',
    'style': {
            'width':'20',
            'height':'20',
            'background-color': '#b3de69',
            'border-color':'black',
            'border-width':'1',
            'shape':'triangle',
            'content': 'data(label)',
            'text-wrap': 'wrap',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
 },

 {
```

```
    'selector': '.Issue',
    'style': {
            'width':'20',
            'height':'20',
            'background-color': '#80b1d3',
            'border-color':'black',
            'border-width':'1',
            'shape':'rectangle',
            'content': 'data(label)',
            'text-wrap': 'wrap',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
},

{
    'selector': '.Method',
    'style':{
            'width':'20',
            'height':'20',
            'background-color': '#ffffb3',
            'border-color':'black',
            'border-width':'1',
            'shape':'ellipse',
            'content': 'data(label)',
            'text-wrap': 'wrap',
```

```
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
},

 {
    'selector': '.Place',
    'style': {
            'width':'20',
            'height':'20',
            'background-color': '#fdb462',
            'border-color':'black',
            'border-width':'1',
            'shape':'vee',
            'content': 'data(label)',
            'text-wrap': 'wrap',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
```

```
            }
},

{
    'selector': '.Publication',
    'style': {
            'width':'20',
            'height':'20',
            'background-color': '#8dd3c7',
            'border-color':'black',
            'border-width':'1',
            'shape':'diamond',
            'content': 'data(label)',
            'text-wrap': 'wrap',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
    },

{
    'selector': '.Literal',
    'style': {
            'width':'20',
            'height':'20',
            'background-color': '#d9d9d9',
            'border-color':'black',
```

```
            'border-width':'1',
            'shape':'concave-hexagon',
            'content': 'data(label)',
            'text-wrap': 'ellipsis',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.2',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
  },

{
   'selector': ':selected',
   'style': {
          'width':'20',
          'height':'20',
          'background-color': '#bc80bd',
          'border-color':'black',
          'border-width':'1',
          'content': 'data(label)',
          'text-wrap': 'wrap',
          'text-max-width': '200px',
          'text-overflow-wrap': 'whitespace',
          'text-valign': 'top',
          'text-background-color': '#FFFFFF',
          'text-background-shape': 'round-rectangle',
          'text-background-opacity': '0.7',
          'font-size': 14,
```

```
                'font-weight':'bold',
                'font-family':'times-new-roman',
                }
            },
        ]

#import json of precomputed positions for entire dataset view
with open('positions.json') as f:
    full_data = json.loads(f.read())



#layout for dash app
app.layout = html.Div([
    html.Div(className='eight columns', children=[



        #the network
          cyto.Cytoscape(
            id='food-pol-net',
            #allow box selection using shift+click & drag
            boxSelectionEnabled= True,
            #initial data = data for all nodes
            elements=full_data,
            #initial layout is precomputed, with spacing factor of 4 for scaling
            layout={
                'name': 'preset','spacingFactor':'4'
            },
            #limit vertical height to allow for tabs
            style={
                'height': '74vh',
                'width': '100%'
            },
            #initially hide Lit Review data
            stylesheet=all_style + dd_style
```

```
    ),
    #tabs
    html.Div(className='four columns', children=[
    dcc.Tabs(id='tabs', children=[

        #Legend for classes, using colour codes and unicode symbols corresponding to shapes
        dcc.Tab(label='Legend', children=[
            html.Div(style=styles['tab'], children=[
                html.P(children=['Classes:',
                html.Div(),
                html.Button('Actor \u2606', style={'background-color': '#fb8072', 'fontSize': 13}),
                html.Button('Action \u25B3', style={'background-color': '#b3de69', 'fontSize': 13}),
                html.Button('Issue \u25A1', style={'background-color': '#80b1d3', 'fontSize': 13}),
                html.Button('Method \u25EF', style={'background-color': '#ffffb3', 'fontSize': 13}),
                html.Button('Place V', style={'background-color': '#fdb462', 'fontSize': 13}),
                html.Button('Publication \u25C7', style={'background-color': '#8dd3c7', 'fontSize': 13}),
                html.Button('Literal/Descriptive \u29D6', style={'background-color': '#d9d9d9', 'fontSize': 13}),
                ]),
                html.P(children=['On Selection:',
                html.Div(),
                html.Button('Selected Node/Edge(s)', style={'background-color': '#bc80bd', 'fontSize': 13}),
                html.Div(),
                html.Button('Edges To Selected Node', style={'background-color': '#fccde5', 'fontSize': 13}),
                html.Button('Edges From Selected Node', style={'background-color': '#bebada', 'fontSize': 13}),
                ])])
            ]),
```

```
        #tab to provide details on demand, in lieu of tooltip functionality
        dcc.Tab(label='Selected Node Info', children=[
            html.Div(style=styles['tab'], children=[
                html.P('Selected Node Data'),
                html.Pre(
                    id='tap-node-data',
                    style=styles['json-output']
                ),

                ])
            ]),

        #tab to provide explore, search and domain-specific pathway functionality, with access to precomputed views
        dcc.Tab(label='Navigation', children=[
            html.Div(style=styles['tab'], children=[
                html.P(children=['Search:',
                dcc.Input(id='search-box', type='text', placeholder='Search Term e.g. NFPOs'),
                html.Button("Search", id='search-button')]),

                html.P(children=['Explore:',
                html.Button("Focus On Selected Node(s)", id='focus-button'),
                html.Button("Expand Selected Node(s)", id='expand-button')]),

                html.P(children=['Load Interaction Pathways:',
                html.Button("Show Actor-Action-Issue Paths", id='path-button'),
                html.Button("Show Actor-Action-Actor Paths", id='aapath-button')]),
                html.P(children=['Load Overviews',
                html.Button("Load Entire Dataset", id='full-button'),
                html.Button("Load Class Hierarchy", id='onto-button'),
                html.Button("Load Class Schema", id='map-button')]),

                ])
            ]),
```

```
        #tab to provide display options, to show or hide nodes/edges
         dcc.Tab(label='Display Options', children=[
           html.Div(style=styles['tab'], children=[


              html.P('Node Display Options:'),
              dcc.RadioItems(
              id='display-radio',
              options=[{'label': 'Hide Academics, Articles, Methods & Publications', 'value': 'dummy-button'},
                       {'label': 'Show All Data', 'value': 'all-button'}], value='dummy-button'),

              dcc.Checklist(
                 id="nodes-checklist",
                 options=[
                 {"label": "Actor", "value": "Actor"},
                  {"label": "Action", "value": "Action"},
                  {"label": "Issue", "value": "Issue"},
                  {"label": "Place", "value": "Place"},
                  {"label": "Publication", "value": "Publication"},
                  {"label": "Method", "value": "Method"},
                  {"label": "Descriptive Info", "value": "Literal"},

                 ],
                 value=['Actor', 'Action', 'Issue'],),

              html.P('Hidden Edges:'),
              dcc.Checklist(
                 id="edges-checklist",
                 options=[
                 {"label": "actionImpacts", "value": "actionImpacts"},
                 {"label": "activeIn", "value": "activeIn"},
                 {"label": "concernsPlace", "value": "concernsPlace"},
```

```
                {"label": "interactWith", "value": "interactWith"},
                {"label": "issueIn", "value": "issueIn"},
                {"label": "locatedIn", "value": "locatedIn"},
                {"label": "publishedIn", "value": "publishedIn"},
                {"label": "relatesTo", "value": "relatesTo"},
                {"label": "takeAction", "value": "takeAction"},
                {"label": "usesMethod", "value": "usesMethod"},
                {"label": "workingOn", "value": "workingOn"},
                {"label": "wrote", "value": "wrote"},
                {"label": "subClassOf", "value": "subClassOf"},
                {"label": "type", "value": "type"},

                ],
                value=[],
                labelStyle={"display": "inline-block"},
            )
            ])
        ]),

        #tab to provide create/manage functionality through adding/deleting data from view
        dcc.Tab(label='Editing Toolkit', children=[
          html.Div(style=styles['tab'], children=[

            html.P(children=['Add Node:', dcc.Input(id='node-name', type='text', placeholder='Name (e.g. Trussell Trust)'),
            dcc.Input(id='node-class', type='text', placeholder='Classes (e.g. Actor, Charity, Group)'),
            html.Button("Add Node", id='node-button')]),

            html.P(children=['Connect Selected Nodes:', dcc.Input(id='edge-name', type='text', placeholder='Edge Label (e.g. worksFor)'), html.Button("Connect Nodes", id='edge-button')]),
```

```
                html.P(children=['Remove Selected Node(s)/Edge(s):',html.Button("Remove Selected", id='remove-button')]),

                ])
            ]),


        ]),
    ]),




])
])

#dash callback to provide interactive functionality
@app.callback((Output('food-pol-net', 'elements'),
               Output('food-pol-net', 'layout'),
               Output('food-pol-net', 'stylesheet')),
              [Input('food-pol-net', 'tapNode'),
               Input('food-pol-net', 'selectedEdgeData'),
               Input('full-button', 'n_clicks'),
               Input('node-name', 'value'),
               Input('node-class', 'value'),
               Input('node-button', 'n_clicks'),
               Input('edge-name', 'value'),
               Input('edge-button', 'n_clicks'),
               Input('onto-button', 'n_clicks'),
               Input('path-button', 'n_clicks'),
               Input('map-button', 'n_clicks'),
               Input('aapath-button', 'n_clicks'),
               Input('display-radio', 'value'),
```

```
            Input("nodes-checklist", "value"),
            Input("edges-checklist", "value"),
            Input('search-button', 'n_clicks'),
            Input('remove-button', 'n_clicks'),
            Input('expand-button', 'n_clicks'),
            Input('focus-button', 'n_clicks'),
            ],
           [State('food-pol-net', 'elements'),
            State('food-pol-net', 'layout'),
            State('food-pol-net', 'stylesheet'),
            State('search-box', 'value'),
            State('food-pol-net', 'selectedNodeData'),
            State('food-pol-net', 'selectedEdgeData'),
            ]
           )
def function(tapnode, tapedge, full_button, nname, nclass, ngen, ename, egen, onto_button, path_button, map_button, aa_path, display, nodes, edges, search_button, remove_button, expand_button, focus, elements, layout, stylesheet, search_term, data_n, data_e):
    #repeat of earlier stylesheets needed within callback function
    dd_style=[
              {
        'selector': '.Academic',
        'style': {
          'display': 'none'
        }
      },

              {
        'selector': '.Article',
        'style': {
          'display': 'none'
        }
      },
```

```
        {
            'selector': '.Method',
            'style':{
                'display': 'none',

                }
            },

    {
        'selector': '.Publication',
        'style': {
            'display': 'none',

        }
    },
    ]

all_style = [
            {
            'selector': 'edge',
            'style': {'width':'1',
                    'content': 'data(label)',
                    'text-rotation':'autorotate',
                    'text-opacity': 0.8,
                    'text-background-shape':'rounded-rectangle',
                    'text-background-color':'white',
                    'text-background-opacity':0.7,
                    'font-family':'helvetica',
                    'font-style':'italic',
                    'font-size': 11,
                    'font-weight':'thin',
                    'target-distance-from-node':'5px',
                    'target-endpoint':'outside-to-node-or-label',
                    'line-opacity': '0.7',
```

```
        'curve-style': 'bezier',
        'control-point-step-size':'100',
        'control-point-weight':0.2,
        'target-arrow-shape': 'vee',
        'arrow-scale': 2,
        'min-zoomed-font-size':16,
        }
},

{
'selector': 'edge[label="equivalentClass"]',
'style': {
      'visibility': 'hidden'
      }
},

{
'selector': 'node[label="Thing"]',
'style': {
      'display': 'none'
      }
},

{
'selector': 'node[label="Nothing"]',
'style': {
      'display': 'none'
      }
},

{
   'selector': '.Actor',
   'style': {
         'width':'20',
```

```
            'height':'20',
            'background-color': '#fb8072',
            'border-color':'black',
            'border-width':'1',
            'shape':'star',
            'content': 'data(label)',
            'text-wrap': 'wrap',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
},

  {
     'selector': '.Action',
     'style': {
             'width':'20',
             'height':'20',
             'background-color': '#b3de69',
             'border-color':'black',
             'border-width':'1',
             'shape':'triangle',
             'content': 'data(label)',
             'text-wrap': 'wrap',
             'text-max-width': '200px',
             'text-overflow-wrap': 'whitespace',
             'text-valign': 'top',
```

```
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
},

{
   'selector': '.Issue',
   'style': {
            'width':'20',
            'height':'20',
            'background-color': '#80b1d3',
            'border-color':'black',
            'border-width':'1',
            'shape':'rectangle',
            'content': 'data(label)',
            'text-wrap': 'wrap',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
},
```

```
 {
    'selector': '.Method',
    'style':{
            'width':'20',
            'height':'20',
            'background-color': '#ffffb3',
            'border-color':'black',
            'border-width':'1',
            'shape':'ellipse',
            'content': 'data(label)',
            'text-wrap': 'wrap',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
},

 {
    'selector': '.Place',
    'style': {
            'width':'20',
            'height':'20',
            'background-color': '#fdb462',
            'border-color':'black',
            'border-width':'1',
            'shape':'vee',
            'content': 'data(label)',
```

```
            'text-wrap': 'wrap',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
},

{
   'selector': '.Publication',
   'style': {
            'width':'20',
            'height':'20',
            'background-color': '#8dd3c7',
            'border-color':'black',
            'border-width':'1',
            'shape':'diamond',
            'content': 'data(label)',
            'text-wrap': 'wrap',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.7',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
```

```
            'min-zoomed-font-size':16,
            }
    },

 {
    'selector': '.Literal',
    'style': {
            'width':'20',
            'height':'20',
            'background-color': '#d9d9d9',
            'border-color':'black',
            'border-width':'1',
            'shape':'concave-hexagon',
            'content': 'data(label)',
            'text-wrap': 'ellipsis',
            'text-max-width': '200px',
            'text-overflow-wrap': 'whitespace',
            'text-valign': 'top',
            'text-background-color': '#FFFFFF',
            'text-background-shape': 'round-rectangle',
            'text-background-opacity': '0.2',
            'font-size': 14,
            'font-weight':'bold',
            'font-family':'times-new-roman',
            'min-zoomed-font-size':16,
            }
 },

{
   'selector': ':selected',
   'style': {
          'width':'20',
          'height':'20',
          'background-color': '#bc80bd',
```

```
                    'border-color':'black',
                    'border-width':'1',
                    'content': 'data(label)',
                    'text-wrap': 'wrap',
                    'text-max-width': '200px',
                    'text-overflow-wrap': 'whitespace',
                    'text-valign': 'top',
                    'text-background-color': '#FFFFFF',
                    'text-background-shape': 'round-rectangle',
                    'text-background-opacity': '0.7',
                    'font-size': 14,
                    'font-weight':'bold',
                    'font-family':'times-new-roman',
                    }
            },
        ]
    #retrieve the context for the triggered state
    ctx = dash.callback_context
    #determine which button was pushed
    button_id = ctx.triggered[0]['prop_id'].split('.')[0]
    #list of all node superclasses
    nodes_list = ['Actor', 'Action', 'Issue', 'Place', 'Publication', 'Method', 'Literal']

    #if the 'load entire dataset' button is pushed, load & display that dataset with precomputed positions
    if (button_id == 'full-button'):
        elements = full_data
        return (elements, {'name': 'preset', 'spacingFactor':'4'},  stylesheet)

    #if the 'load class hierarchy' button is pushed, load & display that dataset with precomputed positions
    if (button_id == 'onto-button'):
        elements = onto_elements
        return (elements, {'name': 'preset'},  stylesheet)
```

```
    #if the 'load class schema' button is pushed, load & display that dataset with precomputed positions
    if (button_id == 'map-button'):
        elements = map_elements
        return (elements, {'name': 'preset'},  stylesheet)

    #if the 'actor-action-issue' path button is pushed:
    if (button_id == 'path-button'):
        #run a SPARQL query to retrieve all entities and interactions linked using the relevant top-level object properties
        qres = g.query(
        """
        SELECT ?actor1 ?gen ?action ?rel ?issue
        WHERE{

       ?action fp:relatesTo ?issue .
       ?action ?rel ?issue .

       ?actor1 fp:takeAction ?action .
       ?actor1 ?gen ?action .
        }
        """ )

        #build a dataframe of the output to enable vectorised application of relevant string methods
        df = pd.DataFrame(columns=['source','interaction','target'])
        for row in qres:
            df= df.append({'source': row.action, 'interaction': row.rel, 'target': row.issue}, ignore_index=True)
            df= df.append({'source': row.actor1, 'interaction': row.gen, 'target': row.action}, ignore_index=True)

        df_out = df.drop_duplicates()
```

```
df_out = df_out[~(df_out['interaction'].str.contains('rdf'))]
df_out = df_out[~(df_out['interaction'].str.contains('owl'))]
df_out = df_out.replace('http://wrenand.co.uk/fpn/', '', regex=True)
df_out = df_out.drop_duplicates()
df_out.replace("", float("NaN"), inplace=True)
df_out.dropna(how='any', inplace=True)
data = df_out

#produce a table of all the new nodes
new_node_table = pd.concat([data['source'], data['target']], axis=0)
new_node_table = new_node_table.drop_duplicates()
node_types = pd.DataFrame(columns=['node', 'class'])

#run a new SPARQL query for EACH node to retrieve a list of all of its classes
for node in new_node_table:
    node = str(node).replace(' ', '_')
    qres = g.query(
        """
        SELECT  ?node_class
        WHERE{
      <http://wrenand.co.uk/fpn/%s> rdf:type ?node_class .
    }
        """ % node)
    #create an empty list of types, fill it with all non-axiomatic types returned by the query above, if there are no returned types it is a Literal.
    type_list = list()
    for row in qres:
        if 'owl' not in str(row.node_class):
            type_list.append(row.node_class.replace('http://wrenand.co.uk/fpn/', ''))
    if len(type_list) == 0:
        type_list.append('Literal')
    node_types = node_types.append({'node' : node, 'class' : type_list}, ignore_index=True)
```

```
        #putting new data into format for visualisation
        type_dict = {}
        for index,row in node_types.iterrows():
            type_dict[row[0]] = row[1]

        new_nodes = list()
        for node in new_node_table:
            new_nodes.append({'data': {'id': str(node), 'label': str(node).replace('_', ' ')}, 'classes':
type_dict[str(node).replace(' ', '_')]})

        new_edges = [
            {'data': {'source': row[0], 'target': row[2], 'label': row[1]}}
            for index, row in data.iterrows()
        ]
        new_elements = new_nodes + new_edges
        #returning these elements, with a compound-spring layout, with current stylesheet.
        return (new_elements, {'name': 'cose-bilkent', 'spacingFactor':'3'}, stylesheet)

    #if the 'actor-action-issue' path button is pushed:
    if (button_id == 'aapath-button'):
        #run a SPARQL query to retrieve all entities and interactions linked using the relevant
top-level object properties
        qres = g.query(
        """
        SELECT ?actor1 ?gen ?action ?imp ?actor2
        WHERE{
       ?actor1 fp:takeAction ?action .
        ?actor1 ?gen ?action .
       ?action fp:actionImpacts ?actor2 .
       ?action ?imp ?actor2 .



        }
        """ )
```

```
#build a dataframe of the output to enable vectorised application of relevant string methods
df = pd.DataFrame(columns=['source','interaction','target'])
for row in qres:
    df= df.append({'source': row.action, 'interaction': row.imp, 'target': row.actor2}, ignore_index=True)
    df= df.append({'source': row.actor1, 'interaction': row.gen, 'target': row.action}, ignore_index=True)

df_out = df.drop_duplicates()
df_out = df_out[~(df_out['interaction'].str.contains('rdf'))]
df_out = df_out[~(df_out['interaction'].str.contains('owl'))]
df_out = df_out.replace('http://wrenand.co.uk/fpn/', '', regex=True)
df_out = df_out.drop_duplicates()
df_out.replace("", float("NaN"), inplace=True)
df_out.dropna(how='any', inplace=True)
data = df_out

#produce a table of all the new nodes
new_node_table = pd.concat([data['source'], data['target']], axis=0)
new_node_table = new_node_table.drop_duplicates()
node_types = pd.DataFrame(columns=['node', 'class'])

#run a new SPARQL query for EACH node to retrieve a list of all of its classes
for node in new_node_table:
   node = str(node).replace(' ', '_')
   qres = g.query(
      """
      SELECT  ?node_class
      WHERE{
     <http://wrenand.co.uk/fpn/%s> rdf:type ?node_class .
   }
      """ % node)
```

```
        #create an empty list of types, fill it with all non-axiomatic types returned by the query above, if there are no returned types it is a Literal.
        type_list = list()

        for row in qres:
            if 'owl' not in str(row.node_class):
                type_list.append(row.node_class.replace('http://wrenand.co.uk/fpn/', ''))
        if len(type_list) == 0:
            type_list.append('Literal')
        node_types = node_types.append({'node' : node, 'class' : type_list}, ignore_index=True)

    #putting new data into format for visualisation
    type_dict = {}
    for index,row in node_types.iterrows():
        type_dict[row[0]] = row[1]

    new_nodes = list()
    for node in new_node_table:
        new_nodes.append({'data': {'id': str(node), 'label': str(node).replace('_', ' ')}, 'classes': type_dict[str(node).replace(' ', '_')]})

    new_edges = [
        {'data': {'source': row[0], 'target': row[2], 'label': row[1]}}
        for index, row in data.iterrows()
    ]
    new_elements = new_nodes + new_edges
    layout = {'name': 'dagre', 'spacingFactor':'5'}

    #returning these elements, with a Dagre layout, with current stylesheet.
    return (new_elements, layout, stylesheet)

#if the search button has been triggered and a term to search has been entered
```

```
    if (button_id == 'search-button') and search_term:

        #compute the similarity scores of this search term against all nodes, and store them in the similarity dataframe
        node_lex['scores'] = np.vectorize(sim_score)(node_lex['node'], search_term)
        #the most similar term is the one with the lowest score (fewest changes as a proportion of length)
        max_sim = node_lex['node'][node_lex['scores'].idxmin()]
        #redefining the search term as the most similar term that exists within the data
        search_term = max_sim
        #preparing term for sparql search
        search_term = search_term.replace(' ', '_')\
        #SPARQL query retrieving all outgoing edges & their connected nodes
        qres = g.query(
        """
        SELECT ?p ?o
        WHERE{
        <http://wrenand.co.uk/fpn/%s> ?p ?o .



        }
        """ % search_term)
        #       #SPARQL query retrieving all incoming edges & their connected nodes
        bqres = g.query(
            """
            SELECT ?s ?p
            WHERE{
            ?s ?p <http://wrenand.co.uk/fpn/%s> .



        }
        """ % search_term)

        #serialising results into dataframe for further use
```

```
    df = pd.DataFrame(columns=['source', 'interaction', 'target'])
    for row in qres:
        df = df.append(
            {'source': search_term, 'interaction': row.p, 'target': row.o}, ignore_index=True)
    for row in bqres:
        df = df.append({'source': row.s, 'interaction': row.p,
                        'target': search_term}, ignore_index=True)
    df_out = df.drop_duplicates()
    df_out = df_out[~(df_out['interaction'].str.contains('rdf'))]
    df_out = df_out[~(df_out['interaction'].str.contains('owl'))]
    df_out = df_out.replace('http://wrenand.co.uk/fpn/', '', regex=True)
    df_out = df_out.drop_duplicates()
    df_out.replace("", float("NaN"), inplace=True)
    df_out.dropna(how='any', inplace=True)
    data = df_out

    #preparing table of all new nodes
    new_node_table = pd.concat([data['source'], data['target']], axis=0)
    new_node_table = new_node_table.drop_duplicates()
    #if no nodes have been returned (i.e. the search term has no connections), return the search
term
    if new_node_table.empty:
        new_node_table = [search_term]

    #produce a table of class lists for all current nodes
    node_types = pd.DataFrame(columns=['node', 'class'])
    for node in new_node_table:
        node = str(node).replace(' ', '_')
        qres = g.query(
            """
            SELECT  ?node_class
            WHERE{
          <http://wrenand.co.uk/fpn/%s> rdf:type ?node_class .
        }
```

```
            """ % node)
        type_list = list()

        for row in qres:
            if 'owl' not in str(row.node_class):
                type_list.append(row.node_class.replace('http://wrenand.co.uk/fpn/', ''))
        if len(type_list) == 0:
            type_list.append('Literal')
        node_types = node_types.append({'node' : node, 'class' : type_list}, ignore_index=True)
    #format data for visualisation
    type_dict = {}
    for index,row in node_types.iterrows():
        type_dict[row[0]] = row[1]
    new_nodes = list()
    for node in new_node_table:
        new_nodes.append({'data': {'id': str(node), 'label': str(node).replace('_', ' ')}, 'classes': type_dict[str(node).replace(' ', '_')]})
    new_edges = [
        {'data': {'source': row[0], 'target': row[2], 'label': row[1]}}
        for index, row in data.iterrows()
    ]
    new_elements = new_nodes + new_edges

    #show all nodes, unless dummy-data-only toggle selected
    stylesheet = all_style
    if (display == 'dummy-button'):
      stylesheet = stylesheet + dd_style

    #From the list of all node superclasses, if that hasn't been selected on the nodes checklist to view - don't display it or any connected edges.
    for node in nodes_list:
        if node not in nodes:
            stylesheet.append({
```

```
            'selector': '.%s' % node,
            'style': {
                'display': 'none'
            }})

        #From the checklist of edge labels, if it has been selected to hide, don't display it
        for edge in edges:
            stylesheet.append({
            'selector': 'edge[label="%s"]' % edge,
            'style': {
                'display': 'none'
            }})

        #display the search term, even if it would be hidden by the display options otherwise
        stylesheet = stylesheet + [{
            'selector': 'node[id="%s"]' % search_term,
            'style': {
                'display': 'element'
            }}]
        #if there are fewer than 200 options, display in a dagre layout, otherwise use compound-spring
        if len(new_elements) < 200:
            layout = {'name': 'dagre', 'spacingFactor':'5'}
        else:
            layout = {'name': 'cose-bilkent', 'spacingFactor':'3'}
        return (new_elements, layout, stylesheet)

    #if the delete button has been pressed, there is data currently displayed, and both node(s) and edge(s) are selected
    if (button_id =='remove-button') and elements and data_n and data_e:
            #produce a list of all nodes and list of all edges that have been selected
            nodes_to_remove = {ele_data['id'] for ele_data in data_n}
            edges_to_remove = {ele_data['id'] for ele_data in data_e}
```

```
        #produce a list of all of the currently displayed data that has not been selected for removal
        intermediate_elements = [ele for ele in elements if ele['data']['id'] not in nodes_to_remove]
        new_elements = [ele for ele in intermediate_elements if ele['data']['id'] not in edges_to_remove]
        #if there are fewer than 200 nodes, display in a dagre layout, otherwise use compound-spring
        if len(new_elements) < 200:
            layout = {'name': 'dagre', 'spacingFactor':'5'}
        else:
            layout = {'name': 'cose-bilkent', 'spacingFactor':'3'}
        return (new_elements, layout, stylesheet)

    #if the delete button has been pressed, if there is data currently displayed, and only node(s) selected
    if (button_id =='remove-button') and elements and data_n:
        # produce a list of all currently displayed nodes not selected for removal
        nodes_to_remove = {ele_data['id'] for ele_data in data_n}
        new_elements = [ele for ele in elements if ele['data']['id'] not in nodes_to_remove]
        #if there are fewer than 200 nodes, display in a dagre layout, otherwise use compound-spring
        if len(new_elements) < 200:
            layout = {'name': 'dagre', 'spacingFactor':'5'}
        else:
            layout = {'name': 'cose-bilkent', 'spacingFactor':'3'}
        return (new_elements, layout, stylesheet)

    #if the delete button has been pressed, if there is data currently displayed, and only edge(s) selected
    if (button_id =='remove-button') and elements and data_e:
        edges_to_remove = {ele_data['id'] for ele_data in data_e}
        new_elements = [ele for ele in elements if ele['data']['id'] not in edges_to_remove]
        if len(new_elements) < 200:
```

```
            layout = {'name': 'dagre', 'spacingFactor':'5'}
        else:
            layout = {'name': 'cose-bilkent', 'spacingFactor':'3'}
        return (new_elements, layout, stylesheet)

    #if the expand button has been pressed, there is data displayed and node(s) selected
    if (button_id =='expand-button') and elements and data_n:
        df = pd.DataFrame(columns=['source', 'interaction', 'target'])
        #insert the current data into the dataframe for results to avoid duplicating currently displayed data later
        for ele in elements:
            if 'source' in ele['data']:
                df = df.append({'source':ele['data']['source'], 'interaction': ele['data']['label'], 'target':ele['data']['target']}, ignore_index=True)

        #run two SPARQL queries for each selected node, to retrieve all outgoing and incoming edges and their connected nodes
        #Lexical similarity not needed here because a node is selected, so exact match guaranteed.
        node_to_expand = {ele_data['id'] for ele_data in data_n}
        for node in node_to_expand:
            search_term = node
            qres = g.query(
            """
            SELECT ?p ?o
            WHERE{
            <http://wrenand.co.uk/fpn/%s> ?p ?o .
            }"""% search_term)

            bqres = g.query(
                """
            SELECT ?s ?p
            WHERE{
           ?s ?p <http://wrenand.co.uk/fpn/%s> .
           }
```

```
""" % search_term)
#also retrieve class membership triples
class_qres = g.query(
"""
SELECT ?s
WHERE{
?s a <http://wrenand.co.uk/fpn/%s> .
}
""" % search_term)

#if each of these queries returns results, add them to the dataframe
if len(qres) > 0:
    for row in qres:
        df = df.append(
            {'source': search_term, 'interaction': row.p, 'target': row.o}, ignore_index=True)
if len(bqres) > 0:
    for row in bqres:
        df = df.append({'source': row.s, 'interaction': row.p,
                'target': search_term}, ignore_index=True)
if len(class_qres) > 0:
    for row in class_qres:
        df = df.append({'source': row.s, 'interaction': 'Type',
                'target': search_term}, ignore_index=True)

#remove namespaces
df_out = df.replace('http://wrenand.co.uk/fpn/', '', regex=True)
df_out = df_out.replace('http://www.w3.org/2002/07/owl#', '', regex=True)
df_out = df_out.replace('http://www.w3.org/1999/02/22-rdf-syntax-ns#', '',
regex=True)
df_out = df_out.replace('http://www.w3.org/2000/01/rdf-schema#', '', regex=True)
df_out = df_out.replace('http://www.w3.org/2001/XMLSchema#', '', regex=True)
#remove reflexive edges
```

```
df_out = df_out[(df_out['source'] != df_out['target'])]
#remove triples that state that something is a class
df_out = df_out[df_out['target'] != 'Class']
#remove all duplicated data (including data that was already in view before pressing expand)
df_out = df_out.drop_duplicates()
df_out.replace("", float("NaN"), inplace=True)
df_out.dropna(how='any', inplace=True)
data = df_out

#produce a table of all new nodes (which includes the old nodes), list of their classes, and format for visualisation
new_node_table = pd.concat([data['source'], data['target']], axis=0)
new_node_table = new_node_table.drop_duplicates()
node_types = pd.DataFrame(columns=['node', 'class'])
for node in new_node_table:
    node = str(node).replace(' ', '_')
    qres = g.query(
        """
        SELECT  ?node_class
        WHERE{
      <http://wrenand.co.uk/fpn/%s> rdf:type ?node_class .
    }
        """ % node)
    class_qres = g.query(
        """
        SELECT  ?node_class
        WHERE{
      <http://wrenand.co.uk/fpn/%s> rdfs:subClassOf ?node_class .
    }
        """ % node)

    type_list = list()
    for row in qres:
```

```
                if 'owl' not in str(row.node_class):
                    type_list.append(row.node_class.replace('http://wrenand.co.uk/fpn/', ''))
            for row in class_qres:
                if 'owl' not in str(row.node_class):
                    type_list.append(row.node_class.replace('http://wrenand.co.uk/fpn/', ''))
            if len(type_list) == 0:
                type_list.append('Literal')
            node_types = node_types.append({'node' : node, 'class' : type_list}, ignore_index=True)
        type_dict = {}
        for index,row in node_types.iterrows():
            type_dict[row[0]] = row[1]

        new_nodes = list()
        for node in new_node_table:
            new_nodes.append({'data': {'id': str(node), 'label': str(node).replace('_', ' ')}, 'classes': type_dict[str(node).replace(' ', '_')]})
        new_edges = [
            {'data': {'source': row[0], 'target': row[2], 'label': row[1]}}
            for index, row in data.iterrows()
        ]

        elements = new_nodes + new_edges
        #recalculate the layout based on the number of nodes now displayed
        if len(elements) < 200:
            layout = {'name': 'dagre', 'spacingFactor':'5'}
        else:
            layout = {'name': 'cose-bilkent', 'spacingFactor':'3'}
    return (elements, layout, stylesheet)

#if the focus button is pressed and node(s) are selected
if (button_id =='focus-button') and data_n:
    df = pd.DataFrame(columns=['source', 'interaction', 'target'])
```

```
#run 3 SPARQL queries on each selected node to retrieve all connected nodes.
#Lexical similarity not needed here because a node is selected, so exact match guaranteed.
node_to_focus = {ele_data['id'] for ele_data in data_n}
for node in node_to_focus:
    search_term = node
    qres = g.query(
    """
    SELECT ?p ?o
    WHERE{
    <http://wrenand.co.uk/fpn/%s> ?p ?o .
    }"""% search_term)

    bqres = g.query(
        """
    SELECT ?s ?p
    WHERE{
  ?s ?p <http://wrenand.co.uk/fpn/%s> .
  }



    """ % search_term)

    class_qres = g.query(
    """
    SELECT ?s
    WHERE{
    ?s a <http://wrenand.co.uk/fpn/%s> .
    }
    """ % search_term)



    if len(qres) > 0:
        for row in qres:
            df = df.append(
```

```
                {'source': search_term, 'interaction': row.p, 'target': row.o}, ignore_index=True)
        if len(bqres) > 0:
            for row in bqres:
                df = df.append({'source': row.s, 'interaction': row.p,
                        'target': search_term}, ignore_index=True)
        if len(class_qres) > 0:
            for row in class_qres:
                df = df.append({'source': row.s, 'interaction': 'Type',
                        'target': search_term}, ignore_index=True)



        df_out = df.replace('http://wrenand.co.uk/fpn/', '', regex=True)
        df_out = df_out.replace('http://www.w3.org/2002/07/owl#', '', regex=True)
        df_out = df_out.replace('http://www.w3.org/1999/02/22-rdf-syntax-ns#', '',
regex=True)
        df_out = df_out.replace('http://www.w3.org/2000/01/rdf-schema#', '', regex=True)
        df_out = df_out.replace('http://www.w3.org/2001/XMLSchema#', '', regex=True)
        df_out = df_out[(df_out['source'] != df_out['target'])]
        df_out = df_out[df_out['target'] != 'Class']
        df_out = df_out.drop_duplicates()
        df_out.replace("", float("NaN"), inplace=True)
        df_out.dropna(how='any', inplace=True)
        data = df_out

        #retrieve classes for each new node to display
        new_node_table = pd.concat([data['source'], data['target']], axis=0)
        new_node_table = new_node_table.drop_duplicates()
        node_types = pd.DataFrame(columns=['node', 'class'])
        for node in new_node_table:
            node = str(node).replace(' ', '_')
            qres = g.query(
                """
                SELECT  ?node_class
                WHERE{
```

```
            <http://wrenand.co.uk/fpn/%s> rdf:type ?node_class .
          }
             """ % node)
          class_qres = g.query(
             """
             SELECT  ?node_class
             WHERE{
            <http://wrenand.co.uk/fpn/%s> rdfs:subClassOf ?node_class .
          }
             """ % node)

          type_list = list()

          for row in qres:
             if 'owl' not in str(row.node_class):
                type_list.append(row.node_class.replace('http://wrenand.co.uk/fpn/', ''))
          for row in class_qres:
             if 'owl' not in str(row.node_class):
                type_list.append(row.node_class.replace('http://wrenand.co.uk/fpn/', ''))
          if len(type_list) == 0:
             type_list.append('Literal')
          node_types = node_types.append({'node' : node, 'class' : type_list},
ignore_index=True)
        #format data for visualisation
         type_dict = {}
         for index,row in node_types.iterrows():
            type_dict[row[0]] = row[1]

         new_nodes = list()
         for node in new_node_table:
            new_nodes.append({'data': {'id': str(node), 'label': str(node).replace('_', ' ')}, 'classes':
type_dict[str(node).replace(' ', '_')]})

         new_edges = [
```

```
            {'data': {'source': row[0], 'target': row[2], 'label': row[1]}}
            for index, row in data.iterrows()
        ]

        elements = new_nodes + new_edges
        #recompute layout based on number of nodes now displayed
        if len(elements) < 200:
            layout = {'name': 'dagre', 'spacingFactor':'5'}
        else:
            layout = {'name': 'cose-bilkent', 'spacingFactor':'3'}
    return (elements, layout, stylesheet)

#if the node display options are changed
if (button_id == 'nodes-checklist'):
    #set initial stylesheet based on which dataset toggle is currently selected
    stylesheet = all_style
    if (display == 'dummy-button'):
        stylesheet = dd_style + stylesheet
    #hide any edges that are currently selected to be hidden
    for edge in edges:
        stylesheet.append({
        'selector': 'edge[label="%s"]' % edge,
        'style': {
            'display': 'none'
        }})
    #display all of the nodes that are selected to show
    for node in nodes_list:
        if node not in nodes:
            stylesheet.append({
            'selector': '.%s' % node,
            'style': {
                'display': 'none'
            }})
    return(elements, layout, stylesheet)
```

```
#if the edge display options are changed:
if (button_id == 'edges-checklist'):
    #set initial stylesheet based on which dataset toggle is currently selected
    stylesheet = all_style
    if (display == 'dummy-button'):
       stylesheet = stylesheet + dd_style
   #display all of the nodes that are selected to show
    for node in nodes_list:
        if node not in nodes:
            stylesheet.append({
            'selector': '.%s' % node,
            'style': {
                'display': 'none'
            }})
    #hide any edges that are currently selected to be hidden
    for edge in edges:
        stylesheet.append({
        'selector': 'edge[label="%s"]' % edge,
        'style': {
            'display': 'none'
        }})

    return (elements, layout, stylesheet)

#if the create node button is pushed, and there is a name and class(es) entered
if (button_id == 'node-button') and nname and nclass:
    #create a new node id with a random integer to allow for duplication
    new_id = nname.replace(' ', '_') + str(np.random.randint(1000, 9999))
    #use the typed name as the node label
    new_label = nname
    #produce a list of the new node's classes, and format data for display
    new_classes = nclass.split(', ')
    new_node = {'data': {'id': new_id, 'label': new_label}, 'classes': new_classes}
```

```
        #append this new node to the currently displayed data
        elements += [new_node]
        return (elements, layout, stylesheet)

    #if the create edge button is pushed, and exactly two nodes are selected, and a label entered
    if (button_id == 'edge-button') and len(data_n)==2 and ename:
        #set the source and target to the first and second nodes selected chronologically, with the
label as entered
        source = data_n[0]['id']
        target = data_n[1]['id']
        label = ename
        #format this for visualisation and append to currently displayed data
        new_edge = [
            {'data': {'source': source, 'target': target, 'label': label}}]
        elements += new_edge
        return (elements, layout, stylesheet)

    #if the display option is set to hide Lit review data, hide it & anything else selected as hidden
    if (display == 'dummy-button'):
        stylesheet = all_style + dd_style
        for node  in nodes_list:
            if node not in nodes:
                stylesheet.append({
                'selector': '.%s' % node,
                'style': {
                    'display': 'none'
                }})
        for edge in edges:
                stylesheet.append({
                'selector': 'edge[label="%s"]' % edge,
                'style': {
                    'display': 'none'
                }})
```

```
#if the display option is set to show all, hide only nodes and edges selected to be hidden
if (display == 'all-button'):
    stylesheet = all_style
    for node in nodes_list:
        if node not in nodes:
            stylesheet.append({
            'selector': '.%s' % node,
            'style': {
                'display': 'none'
            }})
    for edge in edges:
            stylesheet.append({
            'selector': 'edge[label="%s"]' % edge,
            'style': {
                'display': 'none'
            }})

#if a node is tapped
if tapnode:
    #maintain current styling options
    stylesheet = all_style
    if (display == 'dummy-button'):
        stylesheet += dd_style
    for node in nodes_list:
        if node not in nodes:
            stylesheet.append({
            'selector': '.%s' % node,
            'style': {
                'display': 'none'
            }})
    for edge in edges:
            stylesheet.append({
            'selector': 'edge[label="%s"]' % edge,
            'style': {
```

```
                'display': 'none'
            }})
        #for each of the edges connected to the tapped node
        for edge in tapnode['edgesData']:
            #if the edge starts at the selected node, style it
            if edge['source'] == tapnode['data']['id']:
                stylesheet.append({"selector": 'edge[id= "{}"]'.format(edge['id']),
                "style": {"line-color": '#bebada',
                        'target-arrow-color':'#bebada',
                        'width':'3'}})
            #if the edge ends at the selected node, style it
            if edge['target'] == tapnode['data']['id']:
                stylesheet.append({"selector": 'edge[id= "{}"]'.format(edge['id']),
                "style": {"line-color": '#fccde5',
                        'target-arrow-color':'#fccde5',
                        'width':'3'}})

    #if edge(s) are selected, highlight them
    if data_e:
        for edge in data_e:
            stylesheet.append({"selector": 'edge[id= "{}"]'.format(edge['id']), "style": {"line-
color": '#bc80bd',
                        'target-arrow-color':'#bc80bd',
                        'width':'3'}})
        return (elements, layout, stylesheet)

    return (elements, layout, stylesheet)



#if a node is selected, display that node's label and classes in the 'Selected Node Info" tab
@app.callback(Output('tap-node-data', 'children'),
              Input('food-pol-net', 'tapNode'))
def display_tap_node(data):
```

```
    if data:
        name = data['data']['label']
        node_classes = data['classes'].replace(' ', ', ')
        return 'Selected Node:\n' + name + '\nClasses:\n' + node_classes
    return 'nothing selected'




if __name__ == '__main__':
    app.run_server(debug=False)
```

Appendix G Tools

Ontology development: Protégé

Programming Language: Python 3

Knowledge Graph Creation & SPARQL Querying: RDFLib

Inference: Owlready2

Knowledge Graph Vectorisation: OWL2Vec*

Visualisation: Dash-Cytoscape

Free web hosting: Heroku

Fuller details of all packages, versions etc. are given in requirements.txt on https://github.com/Ollie-K/Domain-Knowledge-Graph-Visualisation

Appendix H Use Case Examples

There are multiple ways of achieving some of these use cases (e.g. Use Case 5). Explicit steps will only be provided for one way of achieving each, to illustrate that it is possible.

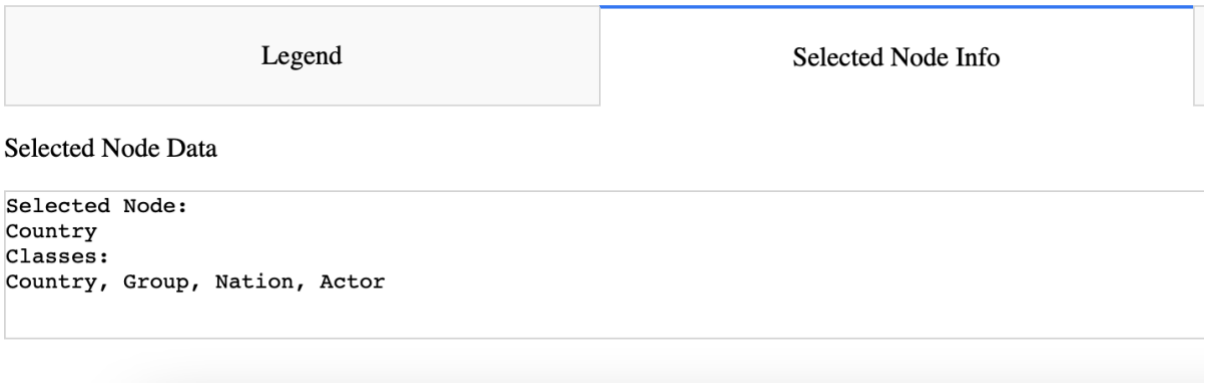
Use Case 1: Visualisation of the entire T-Box.

Use “Show Class Hierarchy” button in the Navigation tab (Fig. 12)

Use Case 2: Visualisation of the information of a class.

1. Use “Show Class Hierarchy” button in the navigation tab.
2. Click a class to select it
3. Use “Selected Node Info” button to display information for this class

Illustrated below with ‘Country’ selected (empty class chosen for clarity).



Use Case 4: Complete visualisation of the A-Box.

Use “Show entire dataset” button with all display options on (Fig. 10a)

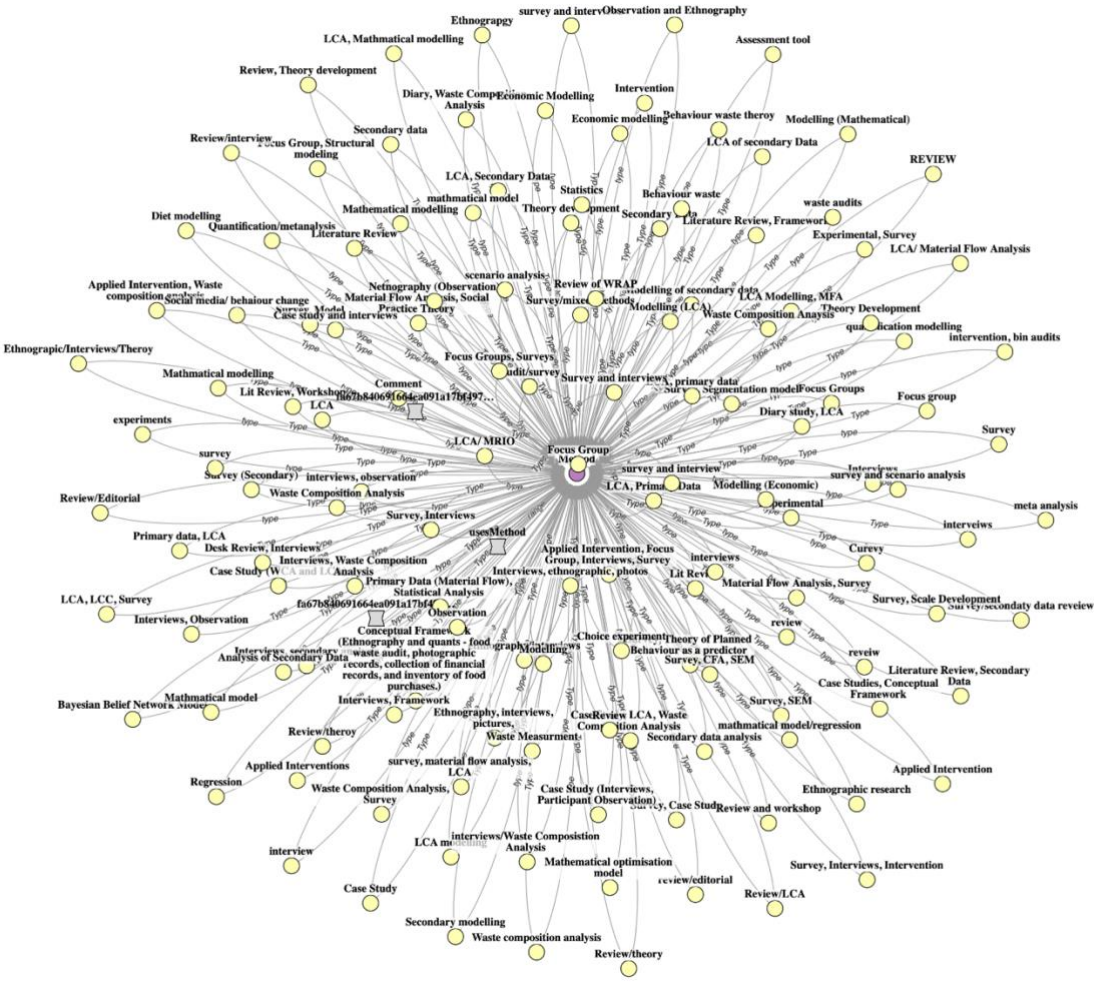
Use Case 5: Visualise the information of an instance

Search for an instance (Fig. 5)

Use Case 6: Show all the Instances of a Class

1. Use “Show Class Hierarchy” button in the navigation tab.
2. Click a class to select it
3. Use ‘Focus on Selected Node” button in the navigation tab.

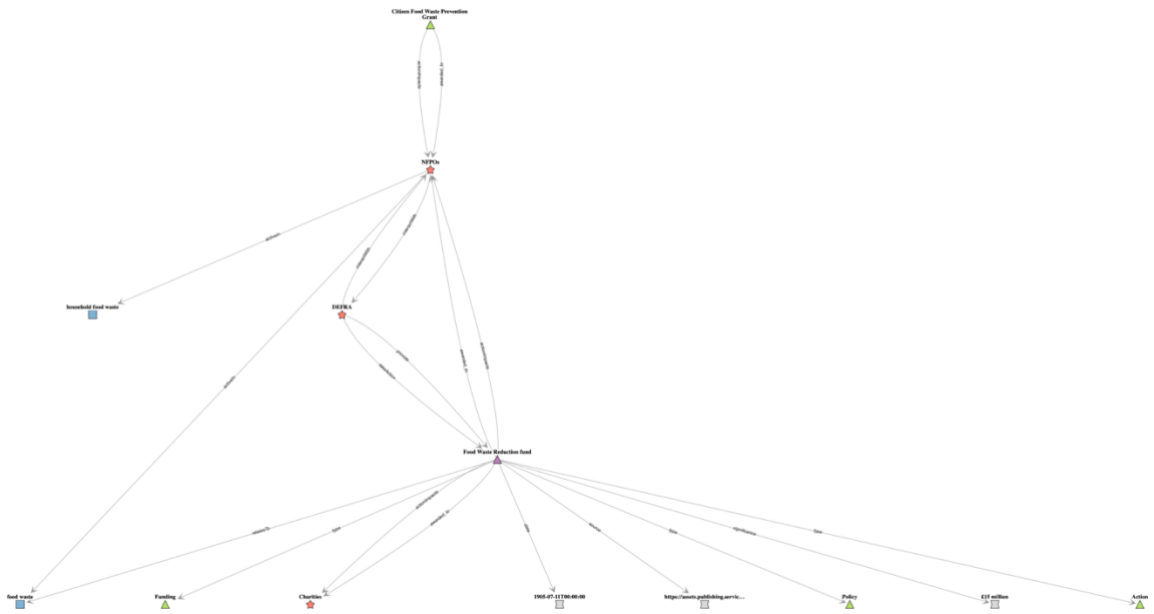
Illustrated below with ‘Method’ selected as class.



Use Case 8: Explore the Dataset

1. Click a node to select it.
2. Click “Expand selected node”

Illustrated below, with navigation starting point of ‘NFPOs’ (Fig. 5), ‘Food Waste Reduction Fund’ has been expanded.



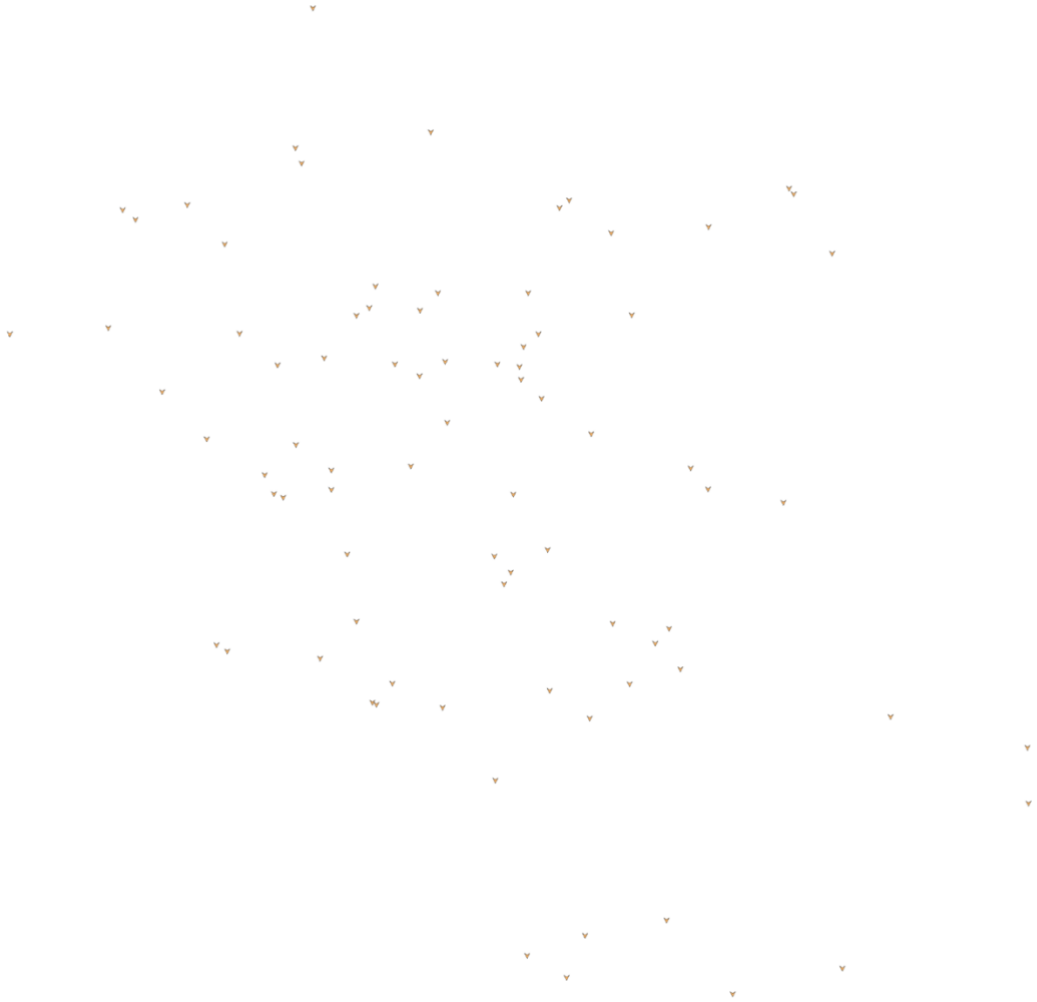
Use Case 11: Create Graphical Visualisation over the Data

Open the system (this visualisation tool is a graphical representation of the data). (Fig. 10b)

Use Case 12: Visualise instances that have specific properties

Use the editing toolkit to adjust which nodes/edges are visualised.

Illustrated below: Full dataset, with only ‘Place’ visible.



Use Case 13: Visualise only the Selected Properties of an Instance

1. Visualise an instance
2. Select unwanted properties (using shift to select multiple or box drag)
3. Delete using the editing toolkit.

Illustrated below using NFPOs (Fig. 5) with both edges and nodes deleted.