## KG Creation: Modelling

***Conceptual Modeling (Task Model)***

Revise the ontology you previously created and apply the basic steps of LOT and evaluate your results using the OntoClean.

**Guidelines:**

**Subtask Model.1:** Motivate your ontology with a set of competency questions or statements and document those in the report. (30%)

**Subtask Model.2:** From the competency questions and statements, derive terms and properties that your ontology needs to cover. Add any missing classes or properties. (30%)

**Subtask Model.3:** Verify the rigidity, identity, and unity of your classes. If changes to your class-hierarchy and/or new properties become necessary, add them to the ontology. (20%)

**Subtask Model.4:** Document all steps and resulting changes to your ontology in the report. (20%)

Competency Questions/Statements:

What is supposed to be in the ontology?

Ontology is about the information of pizza restaurants in America. Therefore, entities such as restaurants, pizza, city, state, etc., as well as the relationships between them, may exist in ontology.

Which concepts are involved in the process of Conceptual Modeling?

Concepts such as restaurant, restaurant menu item, pizza shop, city, state, and country are among the concepts involved in modeling.

What gets to be a class, instance, or property?

The classes we have in the ontology include city, state, and country. Additionally, we have the class "restaurant" to represent the type of location and the class "pizza" to represent the type of menu item required.

The restaurants are connected to their menu items through an object property. They are also linked to the city through another object property and connected to the address and postal code through two other relationships of type data property. On the other hand, the menu items, or pizzas, are linked to the price, currency unit, and description through data properties. Finally, the relationship between the city, state, and country is defined through two object properties.

How do entities, attributes, and relationships relate to each other in Conceptual Modeling?

In conceptual modeling, entities, attributes, and relationships are fundamental components that work together to represent the structure and associations of a domain. Here's how they relate to each other:

1. Entities: Entities represent the real-world objects or concepts within the domain being modeled. They can be concrete, such as a restaurant or an item of menu, or abstract, such as a type of pizza or a category of restaurant. So our entities are items of menu, restaurants, cities, states, countries, pizza types and categories of restaurants

2. Attributes: Attributes define the characteristics or properties of an entity. They describe the specific details or qualities associated with entities. For example, a restaurant entity may have attributes such as name, postcode, and address. Attributes provide further information about the entities they belong to. All of our entities have attribute name but some may have more attributes.

3. Relationships: Relationships establish associations between entities. They define the connections and interactions among entities. Relationships indicate how entities are related and how they interact with each other. For instance, a restaurant may have a relationship with an items of its menus, named has_menu.

What are the different approaches for representing Conceptual Models?

There are several different approaches for representing conceptual models for a knowledge graph of pizza sellers in America. Here are a few common approaches:

1. Graph Database: Use a graph database like Neo4j to model the pizza sellers, restaurants, menu items, cities, states, and countries as nodes in the graph. Relationships can be represented as edges connecting the nodes.

2. RDF/OWL: Use Resource Description Framework (RDF) and Web Ontology Language (OWL) to represent the entities and relationships of the pizza sellers' knowledge graph. RDF allows for expressing the data in triples (subject-predicate-object statements), while OWL provides a more expressive language for modeling the ontology.

3. JSON-LD: Use the JSON-LD (Linked Data) format to represent the pizza sellers' knowledge graph. JSON-LD allows for adding semantics to the JSON data using standardized vocabularies such as Schema.org or custom ontologies.

4. UML Class Diagrams: Use UML (Unified Modeling Language) class diagrams to visually represent the classes (entities) and relationships of the pizza sellers' knowledge graph. Classes can be represented as boxes, and relationships as lines connecting the boxes.

These approaches can be combined or customized based on the specific requirements of the knowledge graph and the tools and technologies being used.

What will the pizza ontology be used for?

The pizza ontology can be used for a variety of purposes related to pizza-related data and knowledge. Some potential uses of the pizza ontology include:

1. Data Integration: The ontology can be used to integrate and organize pizza-related data from different sources, allowing for seamless data integration and retrieval.

2. Knowledge Representation: The ontology provides a formal representation of the concepts, relationships, and properties related to pizzas and pizza restaurants. It allows for structuring and organizing knowledge in a machine-readable format.

3. Querying and Search: With the ontology, it becomes easier to query and search for specific types of pizzas, pizza restaurants, or related information using standardized and structured queries.

4. Recommendations and Personalization: The ontology can be used to power recommendation systems, suggesting specific pizzas or pizza restaurants based on user preferences, dietary restrictions, or other criteria.

5. Data Analytics and Insights: The ontology can serve as a foundation for data analytics, enabling insights and analysis of pizza-related trends, customer preferences, popular toppings, regional variations, and more.

6. Data Interoperability: By following a standardized ontology, data interoperability and exchange between different systems, applications, and platforms can be facilitated.

Overall, the pizza ontology acts as a structured framework for organizing, sharing, and utilizing pizza-related data and knowledge, enabling a range of applications and benefits in the domain.

What types of question shall it answer?

The pizza ontology can answer various types of questions related to pizza restaurants and menu items. Here are a few examples:

1. What are the different types of pizzas offered by a specific restaurant?

2. Which restaurants in a particular city or state offer a specific type of pizza?

3. What is the price of a specific menu item at a particular restaurant?

4. What are the available menu items at a specific restaurant?

5. Which pizza restaurants offer delivery service in a given city?

6. How many pizza restaurants are there in a specific state or country?

7. What are the popular menu items among different pizza restaurants?

8. What are the contact details and addresses of pizza restaurants in a certain area?

These are just a few examples of the types of questions that can be answered using the pizza ontology. The ontology provides a structured and organized representation of knowledge, allowing users to query and retrieve relevant information efficiently.

To reduce redundancy, it has been decided to utilize ontologies provided by Wikidata and Wikipedia to retrieve the entities of city, state, and country. This allows us to invoke those entities through the existing ontologies, minimizing duplication of effort.

Considering the abstract entities, it can be understood that the ontology needs two classes: "pizza_types" and "restaurant category." These classes would help to differentiate between the abstract concept of pizza and restaurant with specific entities and the categories they belong to.

One other consideration is that we are not certain about the currency unit that should be used. Considering that the dataset is about all the pizza shops in America, the currency unit will likely be the US Dollar (USD), and it may not even need to be explicitly mentioned or present in the knowledge graph. However, it has been decided that at least the currency unit should exist as a literal in relation to the menu items, even though it could have been treated as a separate class or entity. For simplicity, it was chosen to represent it as a literal. Two solutions were proposed for adding it to the dataset: one is to include it as part of the restaurant entity in the knowledge graph, and the other is to include it in relation to the restaurant menu items. Although adding it via a property in relation to the menu items might introduce some redundancy in the knowledge graph, it seemed better to follow a similar approach as the price, which is considered in relation to the menu items in the knowledge graph modeling.

Additionally, in a section of the knowledge graph modeling, it was realized that we need different names for menu items as they may overlap with the names of different types of pizzas. For example, many shops may have a menu item called "Margherita Pizza," while "Margherita Pizza" itself is a specific type of pizza and serves as an abstract entity for specifying the type of pizza (which makes querying easier). Therefore, there is a possibility of confusion and incorrect connections if there are identical instances of menu items with the same name. To address this, it was decided to include the name of the restaurant in the menu item name. This will differentiate between the two concepts and make the menu item names unique.

According to the adopted model, the price is considered as a numerical literal specifically associated with pizzas. The other literals, such as postal code, address, currency unit, as well as the names of entities and pizza descriptions, are treated as strings.

Regarding cardinality, each restaurant can be associated with multiple menu items, but each menu item is exclusively associated with only one restaurant. The relationships between the restaurant and the city, city and state, and state and country are all one-to-one relationships.

Each restaurant has only one address and one postal code.

A restaurant can belong to multiple restaurant categories. For example, a restaurant can serve both Italian and American cuisine and therefore be categorized as both Italian and American. However, a pizza item cannot belong to multiple pizza types. Hence, a menu item from a particular restaurant cannot be both a steak pizza and a vegetable pizza. Therefore, the relationship between pizza types and pizza has a cardinality of 1, while the relationship between pizza types and pizza types has an infinite cardinality.

The decision to maintain the rigidity of classes and concepts in the dataset, where there was clear distinction and no shared usage, did not pose any issues. Classes were designed to be rigid, meaning they do not have any subclasses. Only in the case of city, state, and country classes can it be noted that they still maintain rigidity because if a concept such as geographical location were to be a subclass, it would introduce a different scenario that was not accounted for in the knowledge graph model. Therefore, such a class was not included or considered in the ontology.

Indeed, the use of inheritance and subclasses was replaced with abstract relationships to maintain the desired property. This distinction between the modeling stage of the ontology and the implementation stage in constructing the knowledge graph is evident. It highlights the decision-making process and the consideration of specific requirements and constraints for each stage.

To establish identity feature when distinguishing pizzas with the same name, there was indeed an issue. Some shops might have food items listed as "pizza" on their menus, which resulted in a lack of differentiation between instances. However, by creating unique and restaurant-related names for pizza entity instances, the problem was addressed. This ensures that each instance of pizza has a unique identity in the data, minimizing confusion in distinguishing pizzas with the same name. also the property of unity applies to the pizza ontology. In other words, all entities within the ontology are considered "whole entities" where all their parts are related to each other in the same way.