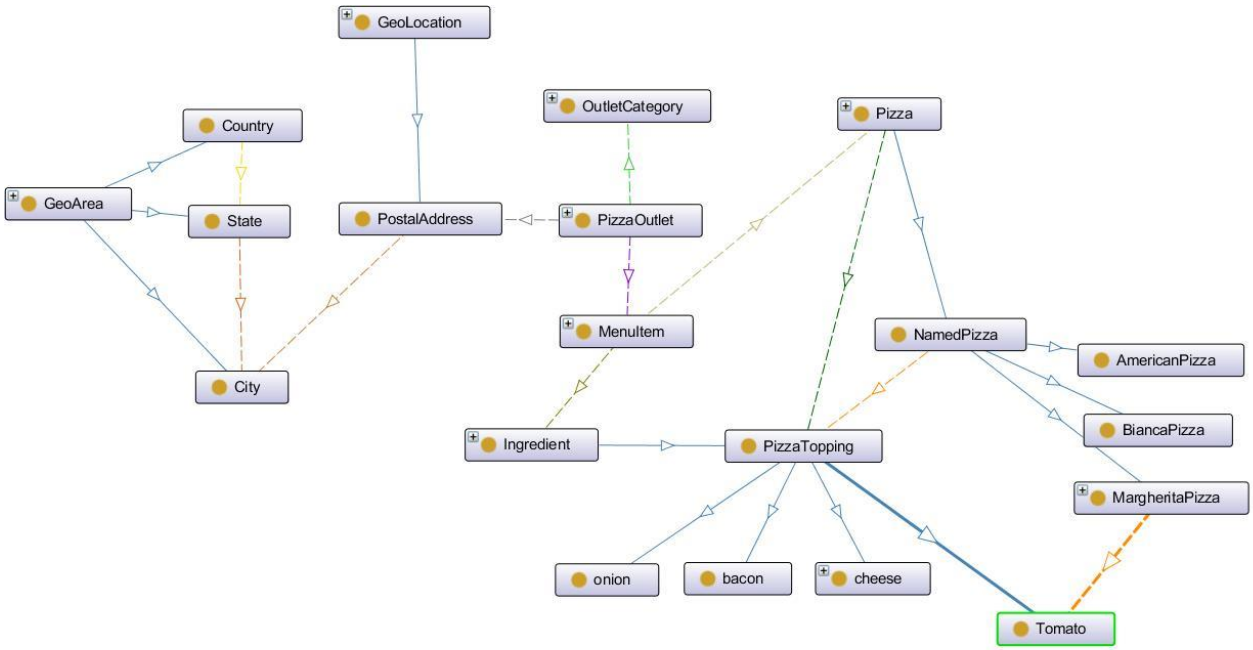
Judith Grieves

INM713 Semantic Web Technologies & Knowledge Graphs - Module Assignment

Introduction

This project uses the Kaggle Pizza dataset [1]. The supplied data is modelled in a Protégé ontology, loaded as RDF triples and aligned with an existing Pizza ontology.

Ontology Modelling (Task OWL)



Ontology Modelling Choices

As the ontology will later be aligned with the Stanford Pizza Ontology [2], care was taken to create and name the generic pizza structures in a similar way to aid the subsequent alignment process.

In general, classes are modelled in such a way as to allow easier future links to other ontologies whilst maintaining simplicity for this project.

Pizza Restaurants

The dataset contains names of places where pizza can be purchased. These businesses are a variety of different types so are modelled as a generic class ***Pizza Outlet***. For each pizza outlet, the data contains a list of outlet types in the categories column. This data is well-defined and finite so could

prove useful in analysing the data. It is modelled as a class ***Category*** to which outlets will be linked by an Object Property ***hasCategory***.

Address and Location Data

The address of the pizza outlet is given as a non-null unstructured text field, mostly containing street name and number. In an ontology where address has a more central role, this could be broken down into more granular structures, but here it is modelled simply as an unstructured string in entity ***Postal Address***. Postcode of the address is also given in the data. It has null values and a variety of formats so is modelled as a string data property of the postal address. Postcode could also be created as a class in an ontology where it has a more critical role.

A postal address identifies a spot geographical location, which could also be identified with latitude and longitude, so a super-class of ***GeoLocation*** is created for possible future integration with external sources. ***Postal Address*** is a sub class of GeoLocation.

For each postal address there is a hierarchy of City, State and Country (where all are populated) so these items will be modelled as a subclasses of ***GeoAreas*** in a class hierarchy, ie. ***City*** > ***State*** > ***Country***.

City is fully populated and will be linked to each Postal Address with the ***cityContainsLocation / addressLocatedIn*** object properties. States are linked to each City via object properties ***stateContainsLocation/cityLocatedIn*** and in turn State will be related to the ***Country*** class (all 'US').

Although State is fully populated in this dataset, there is a mixture of valid 2-char state codes and other address data such as towns, suburbs. For this project, all values in the State columns will be created as State classes to preserve the geographical hierarchy. When resource allows, a more sophisticated exercise should be done to clean the state data and derive values where those given are invalid.

Menu Items

Each row of the dataset contains a menu item sold in a named pizza outlet. These have been modelled as class ***Menu Item.*** 'Item Value' column of the dataset holds (optional) item prices. ***Price*** is only required in numeric calculations so is modelled as a (float) data property of the ***Menu Item*** class along with ***price_currency*** (string).

If the dataset had prices in multiple currencies or they were required in the ontology, price could be modelled as a separate class with its own data properties.

Each menu item in the dataset has a related column 'Item Description'. This is an unstructured text field describing the menu item, from which various items of structured data could be extracted. For information and completeness, the unstructured data will be stored as a data property ***item_description*** of the relevant *Menu Item* and any structured extracts detailed separately, eg. See Ingredient.

Ingredients are an important property of the pizzas and are modelled as a separate class, ***Ingredient,*** linked to the pizza class ***Pizza Topping.***

Pizza Features

The menu items are all types of pizza so a more generalised sub-structure of pizza features is created as ***Pizza***, ***Pizza Topping*** , ***Pizza Base*** and ***Named Pizza Type,*** loosely based on the Stanford Pizza ontology. Where possible, instances will be created from the unstructured data in other columns.

Tabular Data to Knowledge Graph (Task RDF)

Some efforts were made to create a data driven load process with a meta-data file containing program readable details of the classes, object and data properties required. This worked well and allowed amendment of the file to make structural changes. However, there were further elements of data loading required which were not specified in the file, eg. Ingredient and category population and linkages. Defining these entities in a data-driven way could be further developed.

Entity Resolution

To provide the required external links for City, State and Country, we attempt to match instances to elements of the Google Knowledge graph (GKG) [3]. Where a link is established, the instance is created with the Google KG IRIs. We perform a straightforward search of the GKG for each class instance with validation steps depending on class type.

Validations are performed on the returned list of entities. The threshold of validation will determine whether the algorithm will tend to minimise false positives or false negatives. In general, all applications set thresholds in this way dependent on the domain and use of the data.

For this application we only want to link if certain it is correct, ie. avoid false positives. Therefore, each entity returned by the search must have a GKG type of 'Place' and a further specific entity type of City, Administrative Area (State) or Country. Given the more generic nature of the 2 char state codes, we needed to work further down the search results and the number of returned entities to be checked was set to 20. In addition, city search terms must match the entity label to ensure the match is correct. This method successfully identified most valid states and cities and those that were not should be investigated to further improve the quality of the triples created.

Column Transformation Choices

All class instances are created with a label of the original column text for information purposes.

The success of this transformation and the validity of the subsequent SPARQL query results is determined by the quality of the data and to what extent it matches the ontology design. Where there are issues, these will be discussed in this section.

The full pizza dataset contains duplicate rows which should be removed in pre-processing.

Name

The name column contains a textual name of each outlet which will be used to create instances of the class ***Pizza Outlet***, identified by a cleaned, concatenated version of the name string.

Issue: Distinct pizza outlets with duplicate names exist in the data. These are currently created as single instances with multiple addresses but should be resolved to create them with unique URIs.

Address

As described above, each Address column value is created as an instance of class ***Postal Address***. The IRI is a cleaned, concatenated version of the address text.

The Address instance is linked to the appropriate ***City*** class instance by ***addressLocatedIn*** object property.

Postcode is loaded as a string data property ***postcode*** of the appropriate ***Postal Address*** class instance.

City

The City column is fully populated and valid. The values are loaded as instances of the City class and linked to a State with the ***cityLocatedIn*** object property. Cities are matched to GKG to provide entity resolution with an external source.

A significant number of City instances in the CSV exist with different States (eg. Los Angeles). Some are invalid State values and can be validated and handled as described below. There are also valid duplications, eg. Portland, Maine and Portland, Oregon. In this example, Portland Oregon is the first returned entity to pass the City validation steps and its URI is assigned to all instances. The external URI matching algorithm should be enhanced to resolve this issue. Where a City is assigned a local URI, steps must be taken to ensure uniqueness; the addition of State to the URI or assignation of a surrogate key as URI are possible solutions.

State

The ***State*** class instances are currently populated with all values, without validation, and where a link is found to an external IRI, this is used. Each State is linked to the relevant country (US) by creation of ***stateLocatedIn*** object property.

There are a significant number of invalid State instances in the CSV file. These values appear to be suburbs or local areas and there are sufficient cases to warrant a modelling or programmatic resolution. The creation of an 'other' address data entity or property could be used to store invalid State values, to avoid loss of data whilst ensuring that the State instances are all valid. In addition, the external City URI search could also be used to populate a derived State value.

Categories

The categories column contains a well-structured, comma-delimited list of the categories of the pizza outlet. Some simple text cleaning is required to handle other delimiters ('and') and eliminate obvious duplicated data. This list is then be parsed to create a list of unique categories for population of the **Category** class instances.

A search of the ***Category*** class instances for the categories column of each pizza outlet allows the **hasCategory** object property to be created to link a ***Pizza Outlet*** to its Categories.

Menu Item

On loading this column's data to the ***Menu Item*** class instances, duplicates were observed in the naming of pizza menu items across different outlets, eg. 'Cheese Pizza´ at multiple outlets. These instances should not be the same item as different outlets may have different ingredients, price etc.

Possible solutions to this issue include creating an IRI from (a) the concatenation of name + menu item or (b) creating a unique system surrogate key as IRI. This has yet to be implemented.

Country

Always 'US' in this dataset but the code allows for other values and each will be created as an instance of the ***Country*** class and linked to all relevant states and Cities. This class instance is successfully linked to GKG entity 'United States'.

Item value/ Currency

These columns are populated as a ***price*** and ***price_currency*** data properties for their Menu Item class instances.

Item description

As a default, loaded as a description data property of the ***Menu Item***, to be available for information purposes.

After text cleaning, all Item Descriptions can be parsed to create a list of unique ingredients for population of ***Ingredient*** class instances. As a proof of concept in this project, the current transformation has been implemented with a shorter, hard-coded list of ingredients.

After creation of the Ingredient instances, the Description column, for each menu item, is searched for each of the ***Ingredient*** class instances and, where found, the ***hasCategory*** object property is created to link a ***menu item*** to its ***Ingredients***.

SPARQL and Reasoning

The Protégé created ontology and the CSV-generated RDF data were loaded and OWL RL reasoning performed to create an RDF file of the asserted and inferred triples.

34,000 triples were loaded from pizza_full.ttl, increasing to 34,245 after ontology loading and 54,554

after OWL 2 RL reasoning.

SPARQL queries were run against this merged graph.

1. *details of the restaurants that sell pizzas without tomato (i.e., pizza bianca).* The definition of pizza bianca in the OWL ontology did not correctly reason to generate links of the dataset menu items with the NamedPizza class entity so this was not able to be used for this query. Instead I created the query using a simple string search for 'bianca' in the menu items and the query returned 18 rows.
2. *The average price of a Margherita pizza*. As above, the menu items were not successfully linked via equivalence triples to the Margherita Pizza class so this query was executed using regex string matching to find pizzas whose label contained the string 'margherita'. It returned an average price of $12.69.

- *Number of restaurants by city, sorted by state and number of restaurants*. This query returning 800 rows but there were some unreliable results for duplicated data cases in the CSV file, described above, ie. duplicated results for Cities with multiple states in different rows (eg. Los Angeles). This highlights the importance of thorough initial data analysis to inform the modelling and appropriate data wrangling.

3. *List of restaurants with missing postcode.* This query successfully returned 14 restaurants.

Ontology Alignment

Compute Equivalences

The ontology alignment process tries to match entities across different ontologies and, for successful matches, creates triple statements to link matched entities with the owl:equivalentClass or owl:equivalentProperty object properties.

For this exercise, 99 (Stanford) pizza triples were aligned with 27 coursework triples, producing an alignment file of 8 equivalent classes and 6 equivalent properties.

Experiments were carried out using comparison methods, Jaro-Winkler, Levenshtein distance and iSub, and varying threshold values. The best results were obtained In this case, by using the iSub string comparison method between the entity names allowing a match when the score is greater than or equal to 0.9.

The process is currently using a single string comparison. When resource allows, the process could be made more sophisticated with a combination of match algorithms and examining the results.

Perform reasoning

We load (i) the created ontology, (ii) the pizza.owl ontology and (iii) the computed alignment into a graph to perform reasoning. After loading the files there are 2,205 triples and after OWL2 reasoning, 11,247 triples. Unsatisfiable classes are identified by querying the sub-classes of owl:nothing. There are 375 returned by this process needs more understanding and refinement.

Ontology Embeddings

Vector Similarity

OWL2Vec* was run for the created ontology and generated data. Three different configurations were tested, and the vector files analysed below. We chose to vary the sub-graph kernel and walking depth in line with experiments in [4]:

Configuration #	Details
1	Walker=random; walk_depth = 4
2	Walker=random; walk_depth = 3
3	Walker=wl; walk_depth = 3

Similarity results were as follows for the selected word pairs:

Word Pairs	Configuration 1	Configuration 2	Configuration 3
['ingredient', 'topping']	0.6444	0.7781	0.7588
['mozzarella', 'cheese']	0.4108	0.50706	0.3609
['state', 'meatballs']	0.1473	0.44309	0.27949
['state', 'city']	0.3207	0.62135	0.5863
['taverns', 'bars']	0.8464	0.8174	0.69808
['taverns', 'grocery']	0.5183	0.6816	0.5369

For the similarity scores, word pairs were chosen with expected high or low similarity.

As expected, higher similarity scores are achieved for ingredient/topping (0.64 – 0.76), as Topping is a sub-class of Ingredient.

Similarly, ‘taverns and ‘bars’ are both examples of Outlet Category shared by many outlets and achieve a relatively high similarity of between 0.7 and 0.85. By contrast, ‘taverns’ / ‘grocery’ (0.52/0.68) have lower scores. Though both categories, they are unlikely to be found for the same Outlet.

For deliberately dissimilar words, ‘state’ and ‘meatballs’ are chosen. State is a geographical concept in the ontology and ‘meatballs’ an instance of an ingredient in the data. They return a relatively low similarity score of between 0.14 and 0.44.

The difference between the configurations was not extensively analysed although 2 generally achieved higher scores than 1&2.

Subtask Vector.3 Compute clusters

Not completed

References

[1] https://www.kaggle.com/datafiniti/pizza-restaurants-and-the-pizza-they-sell

[2] https://protege.stanford.edu/ontologies/pizza/pizza.owl

[3] https://developers.google.com/knowledge-graph

[4] Chen, J., Hu, P., Jimenez-Ruiz, E., Holter, O.M., Antonyrajah, D. & Horrocks, I. 2020, "OWL2Vec: Embedding of OWL Ontologies", .