

Find and Add Food Recipes using Semantic Technologies

Project Assignment, INFO216

Students: 177

Project description

Food and food recipes are frequently searched topic on the internet, and there exist semantic ontologies for encoding food concepts such as BBC Food Ontology and Recipe Schema from schema.org. But most of these ontologies only focus on the food or recipes themselves. No ontologies (as we found) combines dietary restriction information such as allergens or food glycemic indexes. We want to extend these ontologies to take in information about allergens.

This project takes a dataset, lifts it to semantic level, and allows user interaction through a website.

Dataset

The dataset used for this project can be found in this following link:

<https://github.com/cweber/cookbook/blob/master/recipes.csv>

The dataset contains 60 columns and 229 rows. The dataset are in format of CSV, and the columns are divided in Title, Directions, Quantity01, Unit01, Ingredient01 up to Quantity19, Unit19, and Ingredient19.

I have made some modification to the dataset to better fit our program such as replacement of specific letters and signs, and filling missing cells.

Reading and lifting

The original thinking was to use some program for lifting (for example OpenRefine from Google which converts from CSV to RDF), but later was educated in how to write an own converter in the course.

The written program reads the dataset, applies vocabularies to each cell, and converts it to RDF by adding the data as triples.

The program outputs the converted dataset to a file in Turtle format, which is then applied to Blazegraph.

Tools, Technologies and Vocabularies

The project is mainly written in Python, with some HTML and CSS for the websites. Also, template engine Jinja2 was used as a connection between Python and HTML. The template engine are a component of Flask, a web framework written in Python.

We use Blazegraph to act as a database for the data, and SPARQL as communication link between Python and Blazegraph. Connection to Blazegraph is achieved by using SPARQLWrapper in Python.

We use RDF Vocabulary (<https://www.w3.org/TR/2002/WD-rdf-schema-20020430/>) and Food Ontology (<https://www.bbc.co.uk/ontologies/fo>). The Food Ontology is a simple lightweight ontology for publishing data about recipes, created by BBC. The ontology combines vocabularies from example OWL and RDFS to define their own vocabulary.

Our initial idea was to use Recipe Vocabulary from schema.org (<https://schema.org/Recipe>), but found out that it was not sufficient enough. For example the vocabulary did not have any properties for quantity and units.

System

The program are divided into three components. This splitting might resemble a Model-view-controller design pattern. The file `query_functions.py` acts like a controller between model, which is the Blazegraph, and view, which is the `app.py` and its templates.

`query_functions.py` contains functions that creates and executes queries. The file has a connection to Blazegraph.

Blazegraph holds on the RDF triples. It sends the result to `query_functions`. Blazegraph gets it data from a file called `recipe_triples.txt`. This file is created from `csv_to_rdf.py`.

`app.py` is the program that shows the interface to the user. It uses templates to send and receive results. It connected to `query_functions.py` by using the controller's functions.

The project has also a file for testing queries made in Python called `recipe_queries.py`.

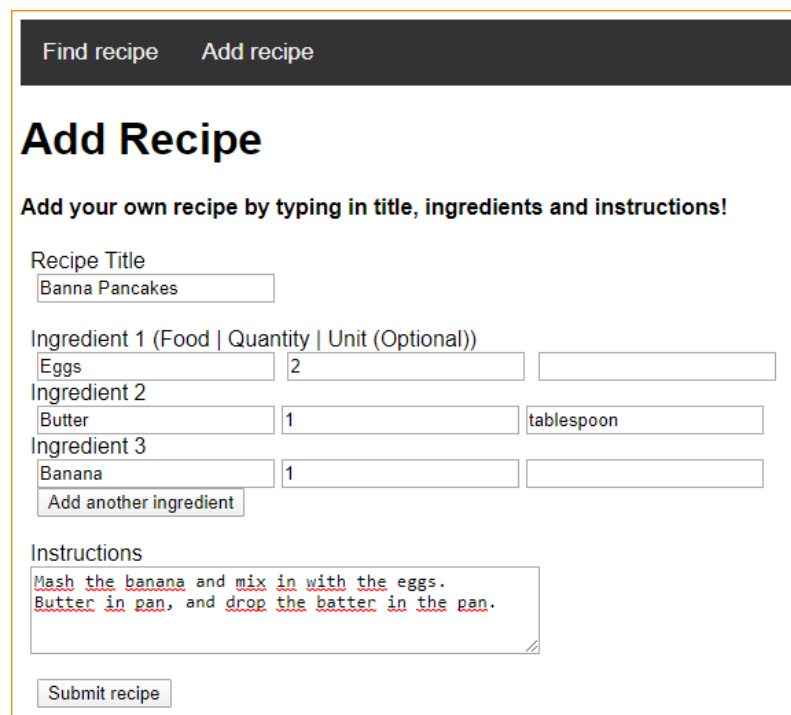
Use cases



The screenshot shows a web interface with a dark header bar containing two links: "Find recipe" and "Add recipe". Below the header, the main heading is "Find recipes". Underneath, it says "Find recipes based on your ingredients!". There are three input fields labeled "Ingredient 1", "Ingredient 2", and "Ingredient 3". The first two fields contain the text "egg" and "salt" respectively. The third field contains "sugar". Below these fields is a button labeled "Add another ingredient". At the bottom of the form is a button labeled "Search for recipes".

Figure 1: Find recipes by typing in ingredients

The user can type in his list of ingredients, and the program will output recipes that matches the user's list of ingredient. The ingredients are in URI, because the intention is to allow the user to click on an ingredient to get more information about it.



The screenshot shows a web interface with a dark header bar containing two links: "Find recipe" and "Add recipe". Below the header, the main heading is "Add Recipe". Underneath, it says "Add your own recipe by typing in title, ingredients and instructions!". There are three input fields labeled "Recipe Title", "Ingredient 1 (Food | Quantity | Unit (Optional))", and "Ingredient 2". The first field contains the text "Banna Pancakes". The second field contains the text "Eggs", "2", and "tablespoon". The third field contains the text "Butter", "1", and "tablespoon". Below these fields is a button labeled "Add another ingredient". At the bottom of the form is a button labeled "Submit recipe".

Figure 2: Add a recipe by filling out the fields

The user can also add his own recipe by typing in title, ingredients and instructions. The program will then add the recipe to the database. Pressing the submit button will send the user to the result page as the same when searching for recipes.

Find recipe

Add recipe

http://example.org/My_Best_Gingerbread
Ingredients:
http://example.org/cinnamon
http://example.org/egg
http://example.org/molasses
http://example.org/butter
http://example.org/salt
http://example.org/soda
http://example.org/sugar
http://example.org/cloves
http://example.org/hot_water
http://example.org/ginger
http://example.org/sifted_flour
Instructions:
Bake in greased pan for 35 minutes at 325-350 degrees.

Figure 3: Result of either finding or adding a recipe

Quality Aspect

Our ontology consists of three classes - Recipe, Food and Ingredient. In our triple-store database there is 229 recipe individuals in addition to insertion from the users.

Each Recipe has up to 19 Ingredients, and each Ingredient are made up of three properties: Food, quantity and imperial-quantity (unit). These three properties are combined into one blank node, which is defined as a completed Ingredient.

Queries

```
1 PREFIX fo: <https://bbc.co.uk/ontologies/fo/>
2 PREFIX ex: <http://example.org/>
3
4 SELECT DISTINCT ?title ?i1 ?i2 ?i3 WHERE
5 {
6
7     ?title fo:ingredients ?ingredient1 .
8     BIND (ex:salt as ?i1)
9     ?ingredient1 fo:food ?i1 .
10
11    ?title fo:ingredients ?ingredient2 .
12    BIND (ex:butter as ?i2)
13    ?ingredient2 fo:food ?i2.
14
15    ?title fo:ingredients ?ingredient3 .
16    BIND (ex:flour as ?i3)
17    ?ingredient3 fo:food ?i3.
18
19 }
```

Figure 4: Example of a SELECT query

The first figure shows a SELECT query. This query gives a result if it finds recipes that matches the three ingredients. First, it find the blank nodes by looking for property fo:ingredients (?ingredient1, ?ingredient2, ?ingredient3). Then, it looks for triples that has the matching URI (?i1, ?i2, ?i3). I used BIND as it was easier to convert a string to an URI in the program.

```
1 PREFIX fo: <https://bbc.co.uk/ontologies/fo/>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX ex: <http://example.org/>
4 INSERT DATA {
5   ex:Nutella_Cake rdf:type fo:Recipe .
6   ex:Nutella rdf:type fo:Food .
7   ex:Nutella_Cake fo:ingredients _:i0 .
8   _:i0 fo:food ex:Nutella .
9   _:i0 fo:quantity 8 .
10  _:i0 fo:imperial_quantity "oz" .
11  _:i0 rdf:type fo:Ingredient .
12  ex:Banana rdf:type fo:Food .
13  ex:Nutella_Cake fo:ingredients _:i1 .
14  _:i1 fo:food ex:Banana .
15  _:i1 fo:quantity 1 .
16  _:i1 rdf:type fo:Ingredient .
17  ex:Nutella_Cake fo:instruction "Mix together the ingrediens.
18 }
```

Figure 5: Example of an INSERT query

The second figure shows an INSERT query. This query begins by defining the recipe title as a type of class fo:Recipe. After that, it creates blank nodes for each ingredients. The blank nodes are annotated with _:i0 and _:i1. The query adds the necessary information to the blank nodes, and defines the blank nodes as a type of class fo:Ingredient.

It might not be the most complex query, but took the most time to figure it out. The function findRecipes in query_functions.py was time consuming, because I do not know how to “give” a query result to another query. The approach I took was to write two query, compare the results, and create a new result with the combination of the two.

Maintainability

The program are relatively easy to run because of python component Flask. What might be difficult to maintain is the graph. The current problem with the program is that I have to manually upload the file containing the converted dataset to Blazegraph. Also when an user insert a recipe, it will only be saved in Blazegraph, which is locally. The file acts like a database. The ideal thinking is to upload the file automatically after converting, and when an user adds a recipe, the recipe gets added to the file. This functionality is supported by a python module called pymantic from sparql. In the future, I would change the current module, SPARQLWrapper, to this module for better maintaining.

Contribution

Originally, the group consisted of two students. Towards the deadline, we decided that I was going to deliver the project alone. This was due to the other student made no contribution to the project.

Evaluation

The process in this project was slow since our group did not have much knowledge in both semantic technologies and the tools. Much time were spent in familiarizing with the technologies, and because of the missing knowledge of the tools and technologies, the project has some lower degree of achievement. It was on purpose to choose an easier task to at least have a final product.

The plan was also to combine, or atleast add our own data, to the existing dataset with allergens, and create our own semantic vocabulary to define a food allergy. If we managed to implement these, then our plan was to allow the user to filter out food recipes based on allergens. But because of the situation that occurred within our group, this goal was not reached.

The Food Ontology from BBC is a good variations of semantic variables such as Steps and Methods. With time, I could split up the instruction into steps to give the instruction a semantic lifting.