# **Recipe QAS**

## **Authors: *Samuel Mcmurray, Fathimath Shahira, Zeinab Nazaraghaei***

1. **Introduction**

**The application is a food/recipe question answering and somewhat of a semantic search service in which users can search for information related to a recipe item using Controlled Natural Language(CNL). CNL is used instead of natural language by using keywords that use keywords such as recipeName, type, course, country, and ingredients. This is done to make it easier to develop the application without the need for extensive knowledge in Natural Language Processing(NLP) although this introduces the need for the user to have some knowledge as to how the application operates. Each of the recipe searches can be filtered by the name of the recipe, the ingredients within a recipe, the type of food it is, the course that it applies to, the country of origin and any combination of the aforementioned information. The information that each item presents can vary based on the user's search but will all include the wiki resource, description, name and combination of other features.**

1. **Application Overview**

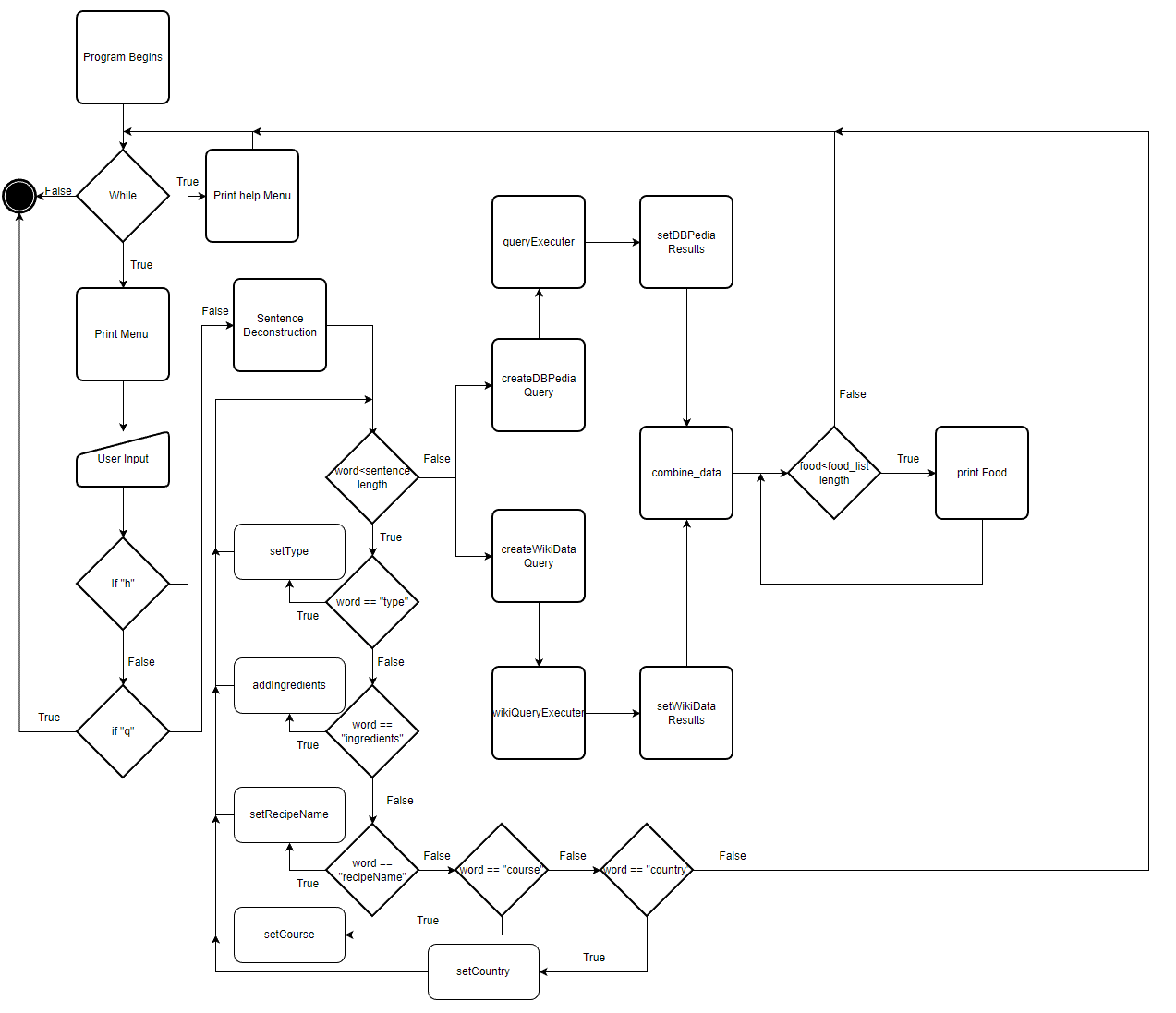
**The application is a python Command-line interface(CLI) that presents 2 main options either the user can access the help menu in which the user is presented with the keywords and how to make use of them in operation. The user can also enter their CNL to run the query accessing both DBpedia and wiki-data 5-star linked open source datasets using the built in ontologies of each. The application utilizes built in functionality of python3 as well as SPARQLWrapper package to handle the queries to the respective databases.**

**Graphical user interface

Description automatically generated**

Figur - RecipeQAS Class Diagram

**The 2 classes that are utilized within the application are Food and QueryBuilder found in the class diagram in Fig. 1, the Food class stores the information gathered from the results of the query. This includes the name of the recipe, description, ingredients, country of origin, course of the recipe, type of food or preparation, url of the dbpedia or the wiki-data resource, image, inventor of the recipe, and the year of the discovery. The QueryBuilder class is used to build the query when a keyword is found the object sets a boolean related to the keyword as well as adding the string that is present after the keyword separated by a semicolon delimiter. In addition the class has the methods to create the queries for each of the databases, this is done by checking whether the booleans are true for the keywords then adds the strings related to the query request to the base query.**

****

Figur - The Application flowchart

**The application as depicted in Fig. 2, begins by entering a while loop that first prints the menu or options available which include to enter ”h” for the help menu, ”q” to exit the application, or the CNL related to the question/search. If the user inputs something other than the listed options it goes to the sentence deconstruction method where the string is split based on delimiters and evaluated as to whether the keyword is present. If the keyword is present the respective query builder method is called to set the value of the string and set the boolean associated to true. Once the loop is finished the query builder methods are called to build the queries. The queries are built by adding the respective strings associated with the information needed to get the results for the particular query and are returned as json. The json results are then set to the food object class where the information is stored then it is combined if the recipe name exists in each the objects are combined returning a new food object list which is then printed to the CLI. Thereafter the loop returns to the beginning.**

**The information available from each of the databases varies, while Dbpedia has better information when it comes to the description of a recipe, the ingredients, the course and a better identifier for type. Dbpedia does not contain information such as the year the recipe was created by the inventor which can be found at wiki-data. Although wiki-data contains valuable information it is quite limited, as such the information from each of the sources is gathered together to make a more complete return on the information that is interesting.**

1. **Results**

**The application can be used to answer a variety of questions related to a recipe, the first case could be that the user is interested in the ingredients of a particular recipe. The user could enter the recipe’s name and leave an empty field for the ingredients in which case the recipe will include a result of the description of the aforementioned recipe along with the ingredients, country of origin, year in which it was created or discovered and an image. In addition if only part of the name is known it can be entered with the results of the related recipes will be presented for instance if one were to enter curry you would get all the curry dishes and the user can locate the one they were thinking of.**

**Another case could be the user has a number of ingredients but is not sure what to make. If the user is interested in something from a particular country they can enter it as well as the ingredients in order to find the dish they can prepare with the ingredients on hand. This can also be the case that the user has an ingredient that is starting to go bad but does not have a recipe in mind. They can enter the ingredient and get a number of dishes that make use of the said ingredient.**

**If the user is interested in a particular course as to what can be prepared such as an appetizer or dessert. If the user enters the course they will get a response with all those recipes that fall under that category. If the user was interested in a certain type of dish they can narrow it down further by including that such as if they were interested in making a cake for dessert or something along those lines.**

**There are many other combinations that a user could come to narrow the search but as with open source data it may be that not all instances are available in every category although if it does exist any number of combinations of the keywords can be used. It is also the case that one instance may exist in one database but not necessarily the other.**

1. **Lessons Learned and Discussions**

**This application gave ample opportunity to learn how to structure queries on multiple databases using SPARQL, how to navigate those databases finding information needed to execute the queries, how the ontologies work within those databases and the challenges associated with working on open source databases. It was interesting learning how to traverse the data in a graph based database using SPARQL, additionally the differences between databases. In the case of wiki-data when building the queries the variables used for the objects did not necessarily have to be used, this made it possible to start with all the variables and then add the queries in as needed. This is in stark contrast to that of Dbpedia where if a variables for the objects was entered an unused an error would occur for a malformed query this added complexity to how the queries could be built. In addition wiki-data followed a very strict format where all answers were objects or properties where an rdfs:label would be found in the object this made it easy to use as all queries would be consistent. Although the wiki-data ontology can be tricky at times due to the tags or ids of the predicates that were associated with each object or property as it was not human readable so it can be confusing when looking at it. On the other hand Dbpedia does not follow any format in some cases it would be a literal and others an object or property, this makes it difficult to get all possible results as not all objects would follow the same format. This can be considered one of the flaws when dealing with these open source databases as some users may use for instance mainIngredient which should be an object but could be a literal or ingredientName which should be literal but can be an object. Although these challenges were present it gave an excellent opportunity to learn more about SPARQL and ontologies as well as linked data sources, it could be considered that as one of the limitations if a strict format is not followed on all cases the it can hamper the use and overall development and deployment to applications.**

# References

Allemang, D., & Hendler, J. (2011). *Semantic Web for the Working Ontologist: Effective Modeling in .* Elsevier Science & Technology.

Brachman, R., & Levesque, H. (2004). *Knowledge Representation and Reasoning.*

Herman, I., Fernández, S., Tejo Alonso, C., & Zakhlestin, A. (n.d.). *SPARQL Endpoint interface to Python*. From SPARQLWrapper: https://sparqlwrapper.readthedocs.io/en/latest/main.html

Prud'hommeaux, E., & Seaborne, A. (2013, March 23). *W3C Reccomendations*. From W3C: https://www.w3.org/TR/rdf-sparql-query/

*Wikidata:SPARQL tutorial*. (2022, December 6). From Wikidata: https://www.wikidata.org/wiki/Wikidata:SPARQL\_tutorial