CSC 751 Project Report: Practical Implementation of OWL Home Ontology

Katarzyna Pasternak, Zishi Wu

May 2020

Abstract

Robotics researchers envision a future in which service robots will assist humans with household tasks. To accomplish this, robotic systems will require robust decision-making software that can explain its intentions to humans, thereby preventing misunderstandings during interactions between humans and robots. In this report, we describe the design a prototype system that combines Natural Language Understanding (NLU) tools with semantic triples in order to create explanations on the relative location of household items, in a way that is easy for humans to understand. This system is a proof-of-concept that Natural Language Understanding and Semantic Web technologies can be integrated together to created an Explainable AI application that runs on the Toyota Human Support Robot platform.

1 Introduction

RoboCup is an annual international robotics competition founded in 1996 with the aim of promoting robotics and AI research, by offering publicly appealing but formidable challenges [1]. The teams in the RoboCup@Home League are challenged with programming service robots such as the Toyota Human Support Robot (HSR) [2] (ref. Figure 1) to assist humans with household tasks (e.g. taking out the trash). To accomplish these tasks, robots require many capabilities such as object detection, object manipulation, navigation, and natural language processing. Robots are expected to work with up to 50 different household items, including brightly-colored paper bags, cereal bowls, and serving trays. The hope is that in the future, service robots will be able to assist the elderly and handicapped with household tasks.

As it is not feasible to work on all capabilities needed by a robot to perform a household task such as taking out the trash, we focused solely on the Natural Language Understanding component of the robot. Our end goal was to design a system that could answer questions regarding the relative location of household items, and present that information in a natural way similar to how a human would respond. To accomplish this, we created an home ontology that contains household item (e.g. fruit) and household location (e.g. kitchen) classes and individuals, as well as relationships describing the location of one object relative to another object (e.g. orange *isLocatedInsideOf* fridge). These household items are a subset of the items specified in the rule book of the RoboCup@Home League [3].

Figure 1: Human Support Robot by Toyota

2 Related Work

Bastianelli et al. [4] used the RoboCup@Home League as a test-bed for gathering speech corpora in interactions between humans and service robots and released their data as a benchmark for spoken language interaction in service robotics.

Berners-Lee [5] proposed the idea of a Semantic Web in which the variations of meaning in human language could be encoded in a standardized format for computers, thereby enabling virtual assistants to understand the nuances in human requests. Hitzler, Krötzsch, and Rudolph detail the foundations of the Semantic Web technologies, which are rooted in ontologies and description logics [6]. The main advantage of an ontology lays in its ability to explain how a software program reached its decision, a phenomena known as *Explainable AI* or **XAI** for short. Anjomshoae et al. [7] define Explainable AI as a domain "aiming at building explainable agents and robots capable of explaining their behavior to a lay user." In this project, we propose to combine ontologies and the RoboCup@Home League challenges to create an Explainable AI system that can enable service robots to explain the location of household items in a way that is natural to humans.

2.1 Semantic Triples

To accomplish this, we used semantic triples to encode information regarding the relative location of household items. A *semantic triple* is a set of three entities — subject, predicate, object — that form a statement about the world. For example, in the statement "Plato is a student of Socrates", the subject is *Plato*, the predicate is *is a student of*, and the object is *Socrates.* Note that the predicate is also referred to as the relationship.

We encoded the relative location of a set of household items into semantic triples. The set of subjects included *orange, banana, mustard, ketchup*, and *bowl.* The set of objects included *fridge, table*, and *chair.* Finally, the set of predicates consisted of five relative location descriptions: *is located on top of*, *is located below*, *is located to left of*, *is located to right of*, *is located inside.*

The statement, "the bowl is on top of the kitchen table" is an example of a fully formed semantic triple that our system can generate as a response to the question, "where is the cereal bowl?" The statement has no ambiguity and the user interacting with the system can quickly go to their refrigerator to check. In contrast, a robotic system that does not have a Natural Language Understanding module that leverages semantic triples may instead respond with raw spatial information from a 2D image. In that scenario, the robot may respond by saying, "the cereal bowl is bounded by pixel coordinates (10, 10), (20, 10), (10, 40), (20, 40) and the kitchen table is bounded by coordinates (0, 40), (100, 40), (0, 80), (100, 80)". The advantage of using semantic triples is that humans find it much easier to understand information on relative location when it is presented in the form of triples, as opposed to raw coordinate data.

2.2 Assumptions

To narrow the scope of the project, we made the following assumptions:

- The robot has already detected the household objects and mapped their locations.
- There is already a script that converts the robot's mapping and object detection data into relative location data.

Under these assumptions, we encoded information on the relative location of household items as semantic triples into an ontology in the home domain.

3 Design of System

The system is divided into two parts: a front-end written in Python that handles natural language understanding and user interaction tasks, and a back-end written in Java that queries the ontology for the location of an object.

3.1 Front-End

First, the Python front-end of the program introduces itself as *Palpi*, the name given to the Toyota Human Support Robot that it runs on. It then prompts the user by asking if they would like to know the location of an item. The user can respond with a question such as "Where is the orange?" A microphone module listens for questions. If the program does not hear a question after about 10 to 15 seconds, it will say "Sorry, I couldn't hear that." and prompt the user again. If the program does hear a question, it will send an audio recording to the Google Cloud Speech-to-Text API and get back a text transcript.

The text is then processed by Rasa, an open-source natural language understanding library for handling tasks such as entity and intent recognition [8]. Rasa uses Tensorflow to train a neural network model on thousands of sentence examples. These sentences are generated by using Chatito, a domain-specific language that allows the user to generate training examples for natural language understanding tasks. In Chatito, users specify a grammar that the training examples generated should follow, with slots for interchangeable words. For example, in the question “where is the orange”, the word *orange* can be interchanged with *mandarin* or *citrus*. Thus in Chatito we would specify in the grammar that the last word in the sentence is a slot with three possibilities: orange, mandarin, and citrus. Chatito also allows us to specify which words refer to entities and which refer to intentions. For example, the word *where* refers to an intention (e.g. inquiry into the location of something) while the word *orange* refers to an entity (e.g. the object that we wish to know the location of). After it is trained on sentence examples generated by Chatito, Rasa can recognize the entity and intention keywords in a sentence.

3.2 Back-End

The keywords recognized by Rasa are sent to a back-end module written in Java that serves as an interface between the user’s questions and the home ontology. The back-end module uses the recognized entities and intentions to build a query in the SPARQL Protocol and RDF Query Language, or SPARQL for short. For example, when a user asks the question “where is the orange”, Rasa will recognize an entity named *orange* and a *location* intention from the *where* keyword. The back-end Java module then generates a SPARQL query from these keywords that selects all the triples where the subject is an orange and the predicate is a type of location relationship (e.g. *isLocatedInside*).

To test our system, we created several individuals in the ontology, including two individuals named *orange* and *fridge*. We defined a semantic triple “orange *isLocatedInside* fridge”, which is returned as a result from executing the SPARQL query and then written to a text file. The front-end then converts this into a more natural format, “orange is located inside the fridge.” Finally, the front-end uses the Google Cloud Text-to-Speech API to convert the text transcript into audio and play the audio response back to the user.

If the SPARQL query failed to find a matching triple in the ontology, then the program will notify the user by saying, “Sorry, I don’t know the location of this object.” If the SPARQL query finds multiple matching triples in the ontology, then the program will say the triples, one by one. For example, if the user asks, “where is the fruit,” the program first says “orange is located inside of fridge,” and then says, “banana is located on top of table.”

3.3 System Outline

Outlined below is a list of steps that details how the program runs:

1. The user says a question to the program: “where is the orange?”
2. The program sends an audio file to the Google Cloud Speech-to-Text service and receives back a text transcript: “where is the orange?”
3. The program passes the text transcription through Rasa’s intent parser to extract entity and intention keywords.
4. The program checks if the entity (e.g. orange) and intention (e.g. location of) keywords exist in the ontology.
5. If the keywords exist in the ontology, the program runs steps 6 through 10. If they don’t exist, the program goes to step 11.

6. The program uses the entity and intention keywords to build a SQARQL query and executes it.

7. The SPARQL query looks for all triples where the subject is an *orange* and the predicate is a sub-class of the relationship class *isLocationOf.* Sub-class relationships in this case include: *isLocatedOnTopOf*, *isLocatedBelow*, *isLocatedToLeftOf*, *isLocatedToRightOf*, *isLocatedInside.*

8. The SPARQL query returns semantic triples that match: *orange isInsideOf fridge.*

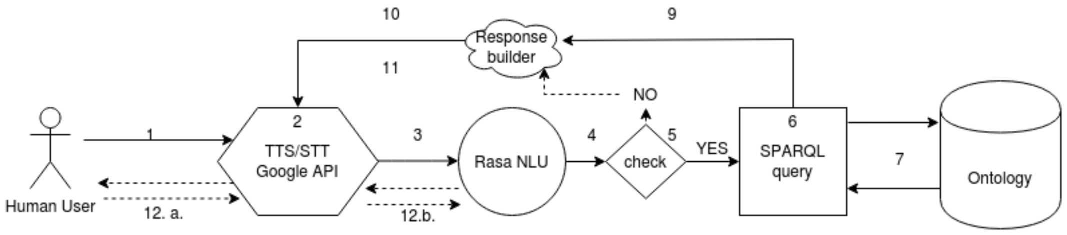
9. The program converts the results of the query into a more natural sentence: "orange is located inside of fridge".

10. The program passes the response text to the Google Text-to-Speech service and gets back an audio file. It plays the audio file, which consists of synthesized speech audio.

11. The program responds by telling the user that the entity they are looking for does not exist in the ontology.

12. Part (a). The program asks the user about the properties of the entity, starting with whether or not the entity belongs to any of the base classes of the ontology (Food, CleaningItem, Furniture). It filters out the individuals that do not belong to that class and then repeats the question, this time with a sub-class of the base class. This exchange continues until the necessary information is collected (i.e. we arrive at a class with no sub-class).

    Part (b). The program ends up with an individual Y in the ontology that belongs to the same class C as the individual X that the user inquired of. It then informs user that while it does not know the location Y, it found an object X that belongs to the same class as Y and suggests that X might be in the same location as Y.



3.4 Limitations

Due to time constraints, we were unable to implement the features described in parts 12 (a) and 12 (b) of the system diagram. We were only able to get the program working for the case where the item does exist in the ontology. If the item does not exist in the ontology, the program simply replies that the item does not exist and asks the user if they would like to ask another question.

4 Dependencies

The following is a list of software tools we used in this project.

- Robot Operating System (ROS), Kinetic version: http://wiki.ros.org/kinetic/Installation
- Ontology Web Language (OWL) API: https://github.com/owlcs/owlapi
- Apache Jena: https://jena.apache.org/
- Protege: https://protege.stanford.edu/
- Rasa NLU: https://github.com/RasaHQ/rasa
- Google Cloud Speech-to-Text API: https://cloud.google.com/speech-to-text/
- Google Cloud Text-to-Speech API: https://cloud.google.com/text-to-speech/
- Chatito: https://github.com/rodrigopivi/Chatito
- Python 2 Speech Recognition API: https://pypi.org/project/SpeechRecognition/

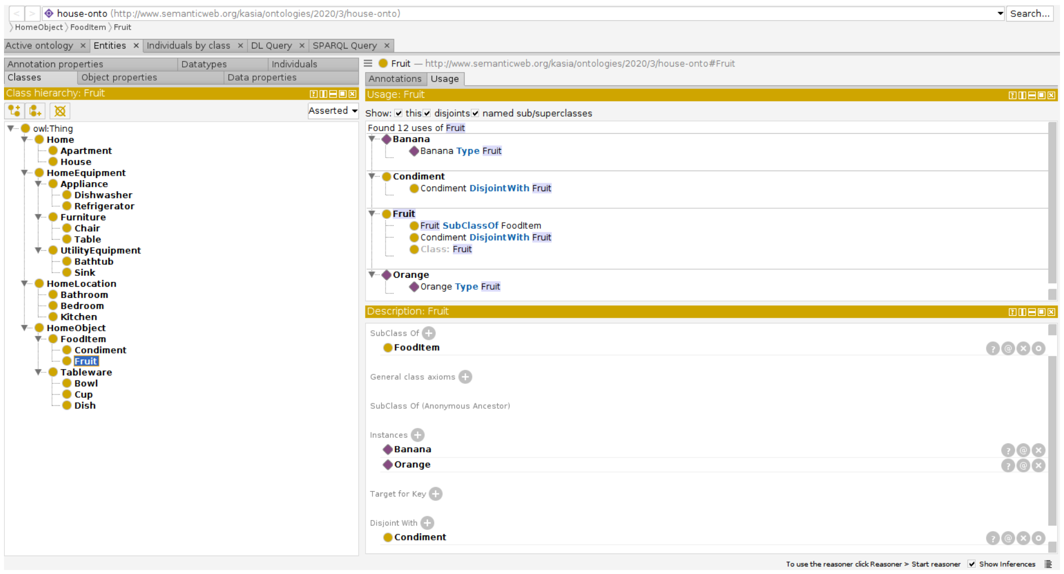
ROS is a set of middleware libraries for robots. It enables users to develop reusable packages and services to handle specific tasks in robotics such as speech recognition, navigation, and mapping. ROS is widely used in both academia and industy, and can operate on various robotics platforms. For this project, we developed a package that allows users to ask a robot questions about the location of an item in a house.

The Ontology Web Language (OWL) API is a Java library for manipulating ontology files. We used Apache Jena to convert OWL ontology files into an RDF data format that can then be queried on by SPARQL queries. To rapidly prototype and test SPARQL queries, we used Protege, an application for creating and validating ontologies.

To convert user questions from speech to text and the program's responses from text to speech, we used services provided by Google Cloud. Chatito was used to generate training examples of questions that the program might expect to hear from users, and the Python 2 Speech Recognition API was used to enable the program to "listen" for the user's questions. The reason why we used Python 2 instead of Python 3, even though Python 2 is no longer being updated as of 2020, is because we are using an older version of ROS called ROS Kinetic, which is written using a mix of C++ and Python 2.

5 Results

At the base of our project, we used the following house ontology:



```
house-onto (http://www.semanticweb.org/kasia/ontologies/2020/3/house-onto)
HomeObject ⟩ FoodItem ⟩ Fruit

Class hierarchy: Fruit                         Asserted
owl:Thing
    Home
        Apartment
        House
    HomeEquipment
        Appliance
            Dishwasher
            Refrigerator
        Furniture
            Chair
            Table
        UtilityEquipment
            Bathtub
            Sink
    HomeLocation
        Bathroom
        Bedroom
        Kitchen
    HomeObject
        FoodItem
            Condiment
            Fruit
        Tableware
            Bowl
            Cup
            Dish

Fruit — http://www.semanticweb.org/kasia/ontologies/2020/3/house-onto#Fruit

Usage: Fruit
Show: ✓ this ✓ disjoints ✓ named sub/superclasses
Found 12 uses of Fruit
Banana
    Banana Type Fruit
Condiment
    Condiment DisjointWith Fruit
Fruit
    Fruit SubClassOf FoodItem
    Condiment DisjointWith Fruit
    Class: Fruit
Orange
    Orange Type Fruit

Description: Fruit
SubClass Of
    FoodItem
General class axioms
SubClass Of (Anonymous Ancestor)
Instances
    Banana
    Orange
Target for Key
Disjoint With
    Condiment

To use the reasoner click Reasoner > Start reasoner   ✓ Show Inferences
```

When running our program, the user observing the terminal can see the following process. First, the user can run the program with the command:

```
(env) kasia@Kasia:~/catkin_ws/src$ rosrun home-ontology GPSCmd_inte
raction_script.py
```

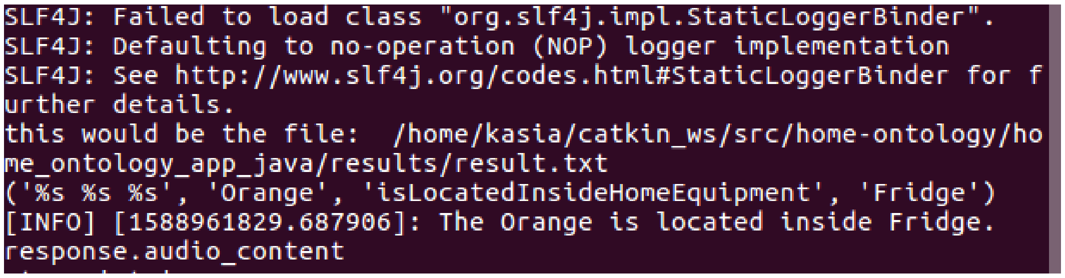
Next, the robot (or the computer running the program) initiates the conversation with the user by introducing itself as "Palpi" and the user what household items they would like the robot to find.

```
[INFO] [1588961808.869467]: Hi, user! My name is Palpi! What would
you like me to find?
response.audio_content
<type 'str'>
Audio content written to file "output.mp3"
```

Then program listens to the user speech and finds the entity and intention keywords. In this case, the intention is *find* and the entity is *fruit*.

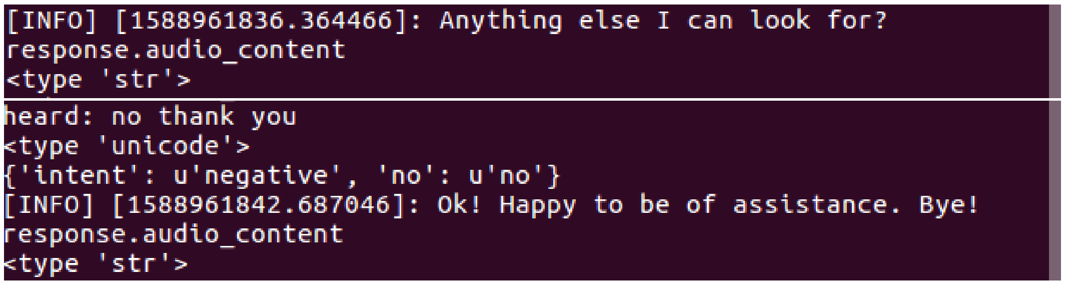
```
<class 'speech_recognition.Microphone'>
heard: hyperpie could you find me a fruit
<type 'unicode'>
{'object': u'fruit', 'intent': u'find'}
```

Then the program executes a SPARQL query on the ontology ontology for given entity and intention pair, and writes the resulting semantic triple(s) in an output file. The output is read from the file, formatted, and synthesized into an audio file that is spoken to the user.



```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for f
urther details.
this would be the file:  /home/kasia/catkin_ws/src/home-ontology/ho
me_ontology_app_java/results/result.txt
('%s %s %s', 'Orange', 'isLocatedInsideHomeEquipment', 'Fridge')
[INFO] [1588961829.687906]: The Orange is located inside Fridge.
response.audio_content
```

Finally, the robot (our machine) asks if there is anything else to find. If the answer is “yes,” then the program repeats the whole process again. If the answer is “no”, then the program finishes executing.



```
[INFO] [1588961836.364466]: Anything else I can look for?
response.audio_content
<type 'str'>
heard: no thank you
<type 'unicode'>
{'intent': u'negative', 'no': u'no'}
[INFO] [1588961842.687046]: Ok! Happy to be of assistance. Bye!
response.audio_content
<type 'str'>
```

6 Future Work

When encountering instances of household items that the ontology has no prior information of, the program currently says “sorry, I don’t know the location of that object” and asks the user if they would like to ask another question. In future work, we would like to make the robot more pro-active by programming it to ask the users about the properties of the item that does not exist in the ontology, and and try to infer the location of the item based on other items that fall in the same category. For example, suppose that the ontology contains the following information

- {milk, cranberry juice} $\in$ (LiquidFood $\cup$ ColdFood)
- {milk, cranberry juice} $isLocatedInsideOf$ fridge
- {LiquidFood, SolidFood, ColdFood, WarmFood} $\subset$ Food

Now suppose the following conversation occurs, where P denotes a person speaking and R denotes the robot speaking. Furthermore, suppose the ontology does not contain information on an individual named *orange juice*.

- P: “Where is the orange juice?”

- R: “I’m not sure but can you tell me about the orange juice?” (Behind the scenes the robot builds a starter question based on the basic classes of the household item ontology; suppose they are: Food, CleaningItem, and Furniture).
- R: “Is orange juice a type of food, cleaning item, or furniture?”
- P: “Orange juice is a type of food.” (The robot can ask a new question based on the sub-classes of Food).
- R: “Is orange juice a type of cold food or warm food?”
- P: Cold food. (The robot rules out instances in the ontology that are not a type of ColdFood).
- R: “Is orange juice a type of liquid food or solid food?”
- P: Liquid food. (The robot rules out instances in the ontology that are not a type of LiquidFood).
- R: “I’m not sure where the orange juice is, but I think it might be in the refrigerator.”
- P: “Why?”
- R: “Well, orange juice is a type of liquid food and cold food. It so happens that milk and cranberry juice also fall into both these categories, and they are located in the refrigerator.”

By using Semantic Web technologies and inference methods, we can give a robot the ability to make an educated guess regarding the location of an item it has not previously detected. In addition to this, Semantic Web technologies enable a robot to explain to a user the reasoning behind why it decided on such a guess. This is all presented in the form of semantic triples, which can encode information on relative location in a format that is easy for humans to understand.

7 Conclusion

In this paper, we presented the design of a prototype system that combines natural language understanding with semantic web triples that answers inquires regarding the relative location of common household items in a clear manner that resembles how humans expect spatial information to be presented. In the future, we plan to extend this technology to handle cases where the ontology does not have information regarding an entity but uses inference rules to suggest the most likely location based on class similarity between entities. As demonstrated, systems containing Explainable AI modules show great promise in service robot applications due to the fact that when asked about their reasoning for a decision, such systems can provide a clear answers in the form of semantic triples. As virtual software and physical robotic assistants are developed for more complex decision-making processes, systems that incorporate Explainable AI will become critical to ensuring safety for humans interacting with robots.

References

[1] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, page 340–347, New York, NY, USA, 1997. Association for Computing Machinery.

[2] Takashi Yamamoto, Tamaki Nishino, Hideki Kajima, Mitsunori Ohta, and Koichi Ikeda. Human support robot (hsr). In *ACM SIGGRAPH 2018 Emerging Technologies*, SIGGRAPH '18, New York, NY, USA, 2018. Association for Computing Machinery.

[3] Luca Iocchi, Dirk Holz, Javier Ruiz-del Solar, Komei Sugiura, and Tijn van der Zant. Robocup@home. *Artif. Intell.*, 229(C):258–281, December 2015.

[4] Emanuele Bastianelli, Luca Iocchi, Daniele Nardi, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. Robocup@home spoken corpus: Using robotic competitions for gathering datasets. In Reinaldo A. C. Bianchi, H. Levent Akin, Subramanian Ramamoorthy, and Komei Sugiura, editors, *RoboCup 2014: Robot World Cup XVIII*, pages 19–30, Cham, 2015. Springer International Publishing.

[5] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[6] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. 01 2009.

[7] Sule Anjomshoae, Amro Najjar, Davide Calvaresi, and Kary Främling. Explainable agents and robots: Results from a systematic literature review. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, page 1078–1088, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems.

[8] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. Rasa: Open source language understanding and dialogue management, 2017.