A Natural Language Control System for Application Specific Robots

Yash Jajoo
Thakur College of Engineering and Technology
University of Mumbai, India

Rutvij Supekar
Vivekanand Education Society's Institute of Technology
University of Mumbai, India

Arrush Hegde
Vellore Institute of Technology
Tamil Nadu, India

ABSTRACT
Due to the increasing presence of robots in several industries, human-robot interaction using natural language has become an important research area. The concept of controlling robots by transforming language instructions in English into executable code for robots is discussed. The proposed approach makes use of semantic similarity by comparing the given instruction with those within a corpus and executes the instruction most similar to the one given. The method is application specific but the modular nature of the system allows it to be adapted for any robot and for any purpose. For this project, a Raspberry Pi based robot following navigation commands is used for experiments and a success rate of 96% is observed.

Keywords
Human-Robot interaction, NLP, LSA, semantic similarity.

1. INTRODUCTION
Robots are becoming highly common for industrial applications. But using robots requires technical expertise and therefore, only those who possess this knowledge are capable of efficiently controlling them. This makes it very important to enable untrained users to interact with them. Since speech is the most natural and common method of communication, it is ideal for controlling robots and if people from all walks of life are to be able to use them, any system for controlling robots must possess the ability of processing human speech. This project explores the concept of using natural language for robots to execute user commands and the main focus of this work is the accurate interpretation of instructions given by users. The system is discussed in the context of controlling robots using human speech interpretation using the example of a Raspberry Pi based robot following navigational instructions. The system takes instructions in English as input, performs semantic analysis and generates the code to be executed by the robot to perform the intended task.

The most important process here is 'semantic analysis' which determines the meaning of the instruction given. Because of that, users are not forced to use a fixed set of words and/or sentences to perform an action and can control a robot using everyday speech. This reduces the complexity of interacting with robots as there is no need to conform to a restricted vocabulary set. This feature makes this method well suited to be used by people who do not possess knowledge on robotics and automation.

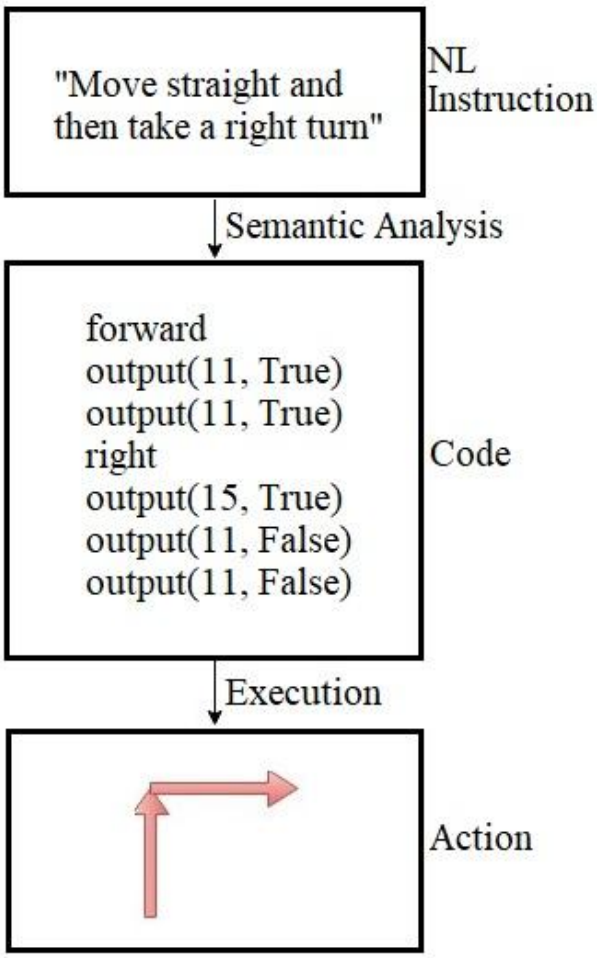


Figure 1: Interpreting NL to generate executable code

The remainder of the paper is organized as follows. First, related research regarding the use of natural language to control robots is mentioned in section 2. Then in section 3, the working of the system is described and experimental results are provided in section 4. Finally, conclusions are made in section 5 followed by a discussion on future work in section 6.

2. RELATED WORK
Several studies exist regarding the use of natural language to instruct robots. Some pair language and robot actions, then build models for mapping instructions to actions [1, 2]. In another study, robots are enabled to learn actions and the lexicons referring to those actions [3]. Researchers have also made use of semantic parsing. One very popular approach involves learning a parser for mapping natural language

commands to control statements [4]. This work, however, uses limited language constructs and a manually created, fixed lexicon to map instructions over robot actions. The process discussed in this paper differs from these as semantic analysis is performed using the concept of topic modeling. That is, similarity of texts is used to determine the intended action. Unlike most NLP studies, this approach is not reliant on large annotated corpora and doesn't need syntactic marking.

3. TECHNICAL APPROACH

The method includes two processes: 'Normalization' and 'Semantic analysis'.

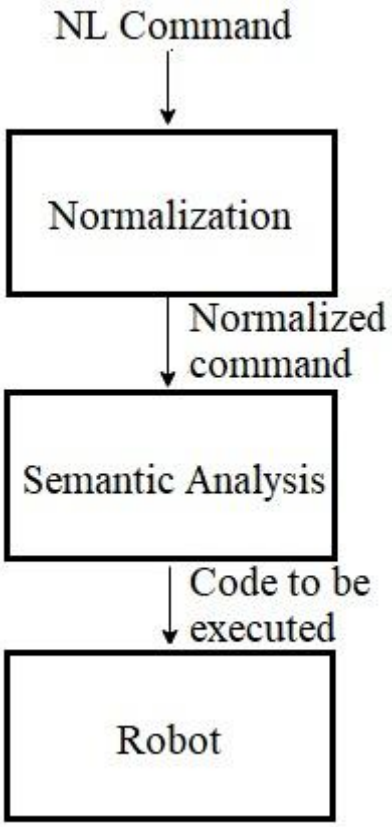


Figure 2: Methodology

3.1 Normalization

Multiple combination of words can be used to issue the same instruction. Here, normalization entails changing the words of the command given to match those of the commands in the corpus. For example, after normalization, the command "Go ahead, turn left, go right and then move straight" will become "Go forward, turn left, go right and then move forward". All the commands in the corpus used here have the words 'forward', 'back', 'right' and 'left' for corresponding actions. The command entered is first normalized and then is treated as input for the following process, semantic analysis.

3.2 Semantic Analysis

The proposed approach makes use of the python library "gensim" [5] short for "generate similar". After importing the corpus, the commands are tokenized and all common as well as non-repeating words are removed. A document representation called 'bag of words' is used and each command in the corpus is represented by a vector whose every element represents a question-answer pair. For example, "Number of times the word 'right' occurs in the document? Ten". A dictionary is then created, assigning unique numbers to all words within the corpus. Each command is represented by a 4-D vector and these tokenized commands are then converted to 'sparse vectors' which are further transformed by Latent Semantic Analysis (LSA) [6]. Cosine similarity is used for determining vector similarity. The normalized command is treated as the input vector which is compared to the corpus vectors for determining semantic similarity. The command in the corpus that is associated with the most similar vector is executed.

3.3 Corpus

The corpus is composed of actions the robot will perform to reach one point on the grid from the other and includes 50 such commands. As mentioned earlier, the corpus is structured in such a way that the words 'forward', 'back', 'right' and 'left' are used for moving ahead, back, right and left respectively and therefore, it includes 4 unique features.

3.4 Example

The assumption here is that the robot is aware of its surroundings and it performs one of the actions specified by the command until it reaches an intersection and/or a dead end. After that, it performs the next action according to the command given.

The functions 'forward()', 'right()', 'left()' and 'back()' are defined for Raspberry Pi based robot for going forward, turning right, turning left and going backwards respectively. Pin 11 controls the motor of the front-left wheel while Pin 15 controls the motor of the front-right wheel. Pin 13 controls the motor of the rear-left wheel while Pin 37 controls the motor of the rear-right wheel. For that, the 'RPi.GPIO' [7] library has been used.

```
GPIO.setup(37, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
GPIO.setup(13, GPIO.OUT)
GPIO.setup(15, GPIO.OUT)
```

(a)

```
def forward():
    GPIO.output(11, True)
    GPIO.output(15, True)

def back():
    GPIO.output(37, True)
    GPIO.output(13, True)

def right():
    GPIO.output(15, True)
    GPIO.output(11, False)
    time.sleep(0.8)
    GPIO.output(15, False)

def left():
    GPIO.output(11, True)
    GPIO.output(15, False)
    time.sleep(0.8)
    GPIO.output(11, False)
```

(b)

Figure 3: (a) GPIO setup for robot's motors (b) Functions for moving the robot

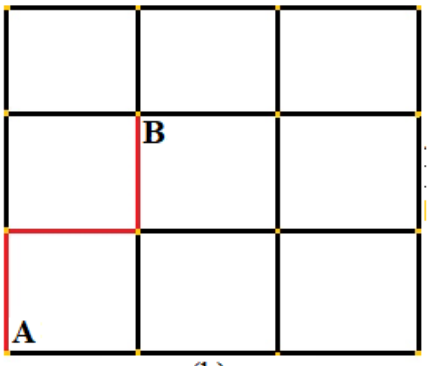
Command: "**Go straight turn right and then take a left**." The robot goes forward, takes a right turn on reaching an intersection and then turns left upon arriving at another intersection.

Executed command: **forward() right() left()** (in that order).

```
setmode ('BOARD',) {}
setup (37, 'OUT'){}
setup (11, 'OUT'){}
setup (13, 'OUT'){}
setup (15, 'OUT'){}
forward
output (11, True){}
output (15, True){}
```

```
right
output (15, True){}
output (11, False){}
output (15, False){}
left
output (11, True){}
output (15, False){}
output (11, False){}
```

(a)



(b)

A: Start Point.
B: Destination.
■: Intersection.

Figure 4: (a) Code to be executed (b) Route Trace

4. EXPERIMENTS

Experimentally, the point of interest in this project is whether the robot reaches the destination by exactly following the instructions. Any given destination can be reached via multiple paths but we consider a trial to be successful if and only if the robot follows the route the instructor intended as the goal is to test instruction-following and not navigation. In other words, the sequence of the actions executed is of the essence. The robot begins at a known starting location and orientation; local knowledge of the grid is updated continuously as the robot progresses. It's important to note that the robot's orientation is considered while traversing the route and not that of the user's.

Trial 1: "**Move forward take a right and then keep going**." Here, the robot first moves forward and turns right once it reaches an intersection. It then moves forward till it reaches an intersection again.

Executed command: **forward() right() forward()** (in that order)

```
setmode ('BOARD') {}
setup (37, 'OUT') {}
setup (11, 'OUT') {}
setup (13, 'OUT') {}
setup (15, 'OUT') {}
forward
output (11, True) {}
output (15, True) {}
```

```
right
output (15, True) {}
output (11, False) {}
output (15, False) {}
forward
output (11, True) {}
output (15, True) {}
```

Figure 5: Code to be executed

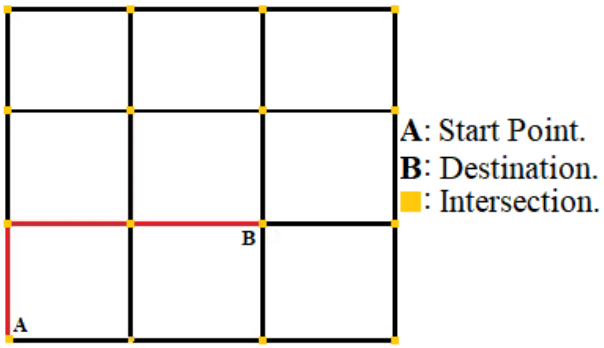


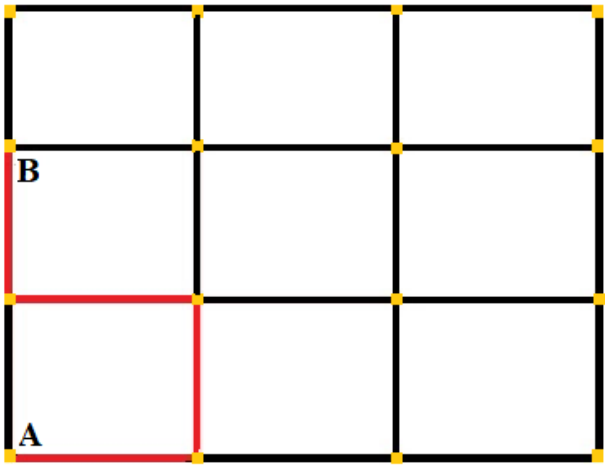
Figure 6: Route Trace

Trial 2: "**Go right and turn left then take another left and turn right**." Here, the robot first moves right until it reaches an intersection and then turns left. Once it reaches an intersection again, it moves left and finally, it takes a right.

Executed command: **right() left() left() right()** (in that order)

```
right                    left
output (15, True) {}     output (11, True) {}
output (11, False) {}    output (15, False){}
output (15, False) {}    output (11, True) {}
left                     right
output (11, True) {}     output (15, True) {}
output (15, False) {}    output (11, False){}
output (11, True) {}     output (15, False){}
```

(a)



A: Start Point.
B: Destination.
■: Intersection.

(b)

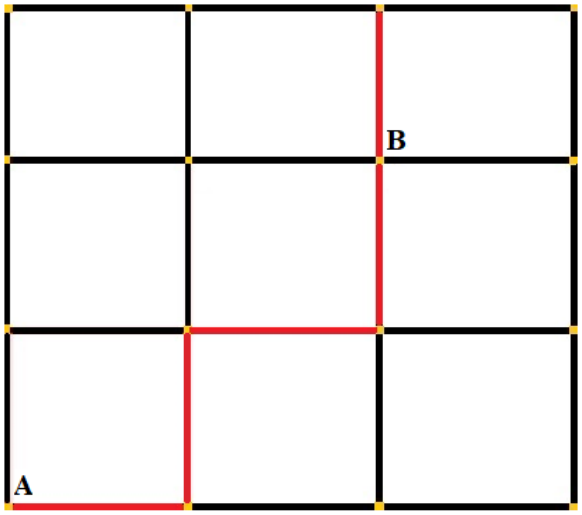
Figure 7: (a) Code to be executed (b) Route Trace

Trial 3: "**Turn right, go left, right and again a left then go straight and come backwards**." Here, the robot moves right and then goes left upon reaching an intersection. Then, it turns right and moves left upon arriving at another intersection. Finally, the robot moves straight and comes back after reaching yet another intersection.

Executed command: **right() left() right() left() forward() back()** (in that order)

right	left
output (15, True) {}	output (11, True) {}
output (11, False) {}	output (15, False) {}
output (15, False) {}	output (11, False) {}
left	forward
output (11, True) {}	output (11, True) {}
output (15, False) {}	output (15, True) {}
output (11, False) {}	back
right	output (37, True) {}
output (15, True) {}	output (13, True) {}
output (11, False) {}	
output (15, False) {}	

(a)



A: Start Point.
B: Destination.
■: Intersection.

(b)

Figure 8: (a) Code to be executed (b) Route Trace

Over the course of this research, a total of 50 experiments were conducted of which 48 were successful. This gives the proposed system a success rate of 96%

5. CONCLUSION

A natural language system for controlling robots is implemented and the same is demonstrated through successfully executing control commands for navigating a robot using natural language instructions as input. The methodology here is fairly simple; it has two sub-processes and can be adapted to suit any robot for any purpose. A total of 50 experiments with non-technical sentences were performed and the system was able to successfully convert spoken sentences into code with 96% success rate. This makes this approach robust enough to be used by laypeople to control robots efficiently. The work reported in this paper provides another step towards expanding the use of natural language in human-robot interaction.

6. FUTURE WORK

To expand on this research, support for multiple languages can be added to make the system feasible for non-English speaking people. Also, the methodology here is demonstrated using a single application, that is, navigating. To be used for any other purpose, modifications specific to that purpose have to be made. So, another research avenue is to enable the approach to be used for multiple applications using a single corpus and without explicit modifications.

7. REFERENCES

[1] S. Tellex et al., "Understanding natural language commands for robotic navigation and mobile manipulation" in Proceedings of the National Conference on Artificial Intelligence (AAAI), Cambridge, MA, 2011.

[2] D. K. Misra et al., " Tell me Dave: context sensitive grounding of natural language to manipulation instructions" in Proceedings of Robotics: Science and Systems, Berkeley, USA, July 2014

[3] L. She et al., "Back to the blocks world: Learning new actions through situated human robot dialogue" in Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), Philadelphia, PA, 2014, pp.89-97.

[4] C. Matuszek et al., "Learning to parse natural language commands to a robot control system" in Proceedings of International Symposium on Experimental Robotics, Quebec City, Canada, 2012.

[5] R.Rehurek and P.Sojka, "Software framework for topic modeling with large corpora" in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta, 2010, pp.46-50

[6] T. Landauer, P. Foltz, and D. Laham, "An Introduction to Latent Semantic Analysis", in Discourse Processes, Taylor and Francis, Nov 2009, pp.259-284

[7] RPi.GPIO 0.6.3. A module to control Raspberry Pi GPIO channels. https://pypi.python.org/pypi/RPi.GPIO