Language-Based Actions for Self-Driving Robot

Dhruv Solanki[1], Mohit Tiwari[2], Aditya Vartak[3], Saurabh Wanivadekar[4], Prof. Naina Kaushik[5]

[1,2,3,4]Student, Dept. Of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India
[5]Professor, Dept. Of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India

---

Abstract - *We present a framework to automatically learn to drive a mobile robot under natural-language command. This framework contains Sequence Modeling (to learn the meaning of sentence and based on sentence try to locate the nouns in environment and identify the associated prepositions) and Cognizance (to generate path and move on that path to accomplish target navigational goal along with an activity in the scenario. This activity can be extinguishing fire, moving objects in the environment etc.*

Key Words: Artificial Intelligence, Natural Language Human-Robot interaction, Robot Simulation, Sequence Modeling, Path finding.

1.INTRODUCTION

An autonomous robot which act and behave with higher degree of independence. In today's changing world it is necessary for an autonomous robot to work on the Natural Language commands of Humans. Autonomous robots are desirable in fields such as space travel, household work like cleaning, waste water treatment and delivering goods and services. A fully autonomous robot can gain information about environment, work for extended period without human intervention, move either all or part of itself throughout its operating environment without human assistance, avoid situations that are harmful to people, property, or itself unless those are part of its design specification, an autonomous robot may also learn or gain new knowledge like adjusting for new methods of accomplishing its tasks or adapting to changing surroundings. Barrett et al [1] proposed a solution for robot navigation using Natural Language and our work is inspired from them. We are going to simulate the framework using Unity Game Engine [2].

Data flow diagram of the figure is given in Fig-1. In this framework the meaning of a sentence that describe path driven by a robot through an environment containing number of objects. An example of such sentence is "Go to the fire which is behind the box and in front of the stool then go towards the table". Such sentences are sequence of description in terms of objects in the environment. Nouns in the descriptions indicate the class of objects involved, such as box, stool, table and fire. Prepositions, such as "in front of" will describe the changing position of robot in time(e.g. in front of the stool),as well as to describe the relative positions of objects in the environment (e.g. the fire which is behind the box).We use Spacy[3] to process and apply Natural Language processing over the sentences to extract nouns and preposition phrases and their target and referent nouns. Sequence Modeling uses large sets of hidden Markov models (HMMs), one for each path sentence pair in training. During training, robot learns meanings of words and phrases. These learned meanings are used in Cognizance phase where the robot drives according to the input sentences. The Implementation details and Results are explained in the following sections.

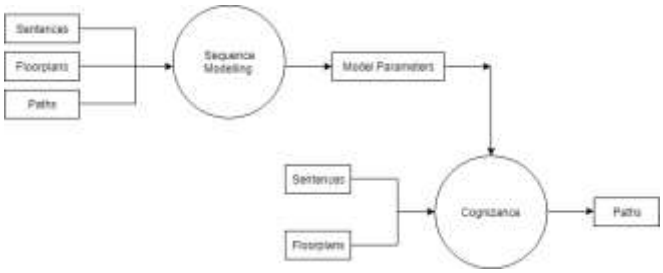

Fig-1: Data Flow diagram

2. CREATING VIRTUAL ENVIRONMENT AND DATA ENGINEERING

2.1 Need for Virtual Environment

Before any system is to be implemented in hardware, it is always a best practice to simulate the system in possible environments to gauge the readiness of deployment in hardware. It is also a safe way of testing the performance and is economical for the organizations on possible failed projects. This is widely followed particularly in the field of Robotics. Hence, we decided to create a virtual environment and robot simulator initially. We chose Unity Game Engine for this task because it is flexible enough for creating virtual environments, collecting data and also supports navigation tools like NavMesh for A* path finding.

2.2 Environment Creation

We created the environment in Unity consisting of a floor or terrain and then placing prefabs of different objects like cone, stool, table etc. on the floor. The Prefab Asset is a template from which you can create new Prefab objects in the Unity Scene. Then we randomized the initializing positions. We then created a robot in the environment

with a view from its view port. Thus, we created an environment with proper objects placed randomly on the floor and a robot which can move around the environment using controls on the host.

2.3 Data Collection

We also set up scripts to make sure raw data i.e. Floorplan, Path and Sentences are to be collected for every run of the environment. The coordinates of objects along with label is collected as floorplan data. The sentences entered in input box is collected as sentences data. We would also move the robot in Unity environment according to entered sentence to capture the path of the robot and thus collect the path data too. We collect total of 3 sentences per path driven by robot. All the three data files of floorplan, sentence and path is passed to server over a network for data preprocessing. The server would have all the framework required for the robot navigation.

2.4 Data Preprocessing

After we obtain raw data from the simulated environment on Unity Engine, we need to make this data suitable to be used and processed for training of the robot. Hence, we do the following steps for the 3 different raw data – sentences, floorplan and path. For Path Data, the collection of path coordinates needs to be preprocessed to remove repeated coordinates. Coordinates may get repeated in consecutive occurrences if robot hasn't moved in the frame of time. This allows us to reduce redundancy of data. For Floorplan, since the floorplan is the distribution of objects in the environment, each floorplan object is captured as a tuple of its location in x-y plane and its class i.e. (x, y, class). Thus, representation of each object becomes understandable.

For Sentences, they are captured from the Input box. We need to be able to extract meaningful sentences from the data and discard meaningless sentences like "This is robot" as they are of no use to the system as they won't contribute to goals of the system. This leaves us with a few types of sentences. The above process could have been done with techniques for sentence classification by Machine Learning Approach. However, this approach is Data-Intensive and requires high Computation cost. Hence, we chose to go with Rule-based Classification. Algorithm 2 is the Rule Based Classification Algorithm. This is because, we identified that the types of sentences for a particular goal are not vast and mostly have structures which can be parsed by grammars. Hence, a grammar can be generated for types of sentences to be encountered for the goal. Some Examples of sentence structures and its corresponding sentences are given in the Table -1.

Table-1: Examples of Sentence Structures found in use of framework to extinguish Fire

Sr. No	Sentence Structure	Example of Sentence
1)	VERB DET NOUN WH-determiner VERB ADP DET NOUN ADP DET NOUN …	"Extinguish the fire which is to the left of the chair"
2)	VERB ADP DET NOUN WH-determiner VERB ADP DET NOUN ADP DET NOUN...	"Go to the fire which is to the left of the stool"

3. SEQUENCE MODELING

We apply Hidden Markov Models (HMM) for each temporal segment and the state in the HMMs are the temporal segments of sentence and the observation of each hidden state is sequence of waypoints. A transition matrix and emission matrix are generated for each state. The transition matrix and emission matrix are passed to forward-backward algorithm to compute the probability of HMM being in each state at each point in path. It also estimates how true the entire sentence is of the path.

3.1 Creating Graphical Model

Once we have broken down the sentence into temporal segments, we need to model them in mathematical form for processing the data. Hence, we convert every segment of sentence into a Probabilistic Graphical Model [4]. Fig-2 shows the graphical model. Then, a we create a floorplan variable (O_1–O_n) for each noun in each segment. Then we apply the label distribution of noun to the variable's set of labels. Finally, we need to use the subject and object of preposition in the sentence, and find each preposition distributions over relative positions and velocities are applied between its the subject and the object. The subject noun act as target object and object noun acts as referent object for the preposition.

Every Probabilistic Graphical Model starts with a root state Pi followed by a preposition or list of prepositions as children. Each preposition is followed by a noun which is its object and preceded by a noun which is its subject for e.g. "fire which is to the left of table" where subject of left is *fire* and object is *table.* Such a preposition is having adjectival use in the sentence. In case of preposition phrases like "towards the bag", there is no need to have a subject noun for it. Just the object would suffice. Such prepositions are called adverbial usage of preposition.

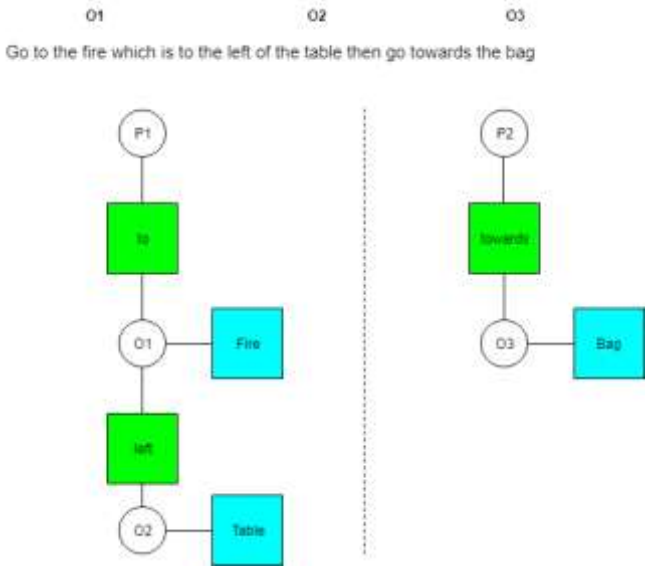

Fig-2: Graphical models created for Sentence. Green box denotes Prepositions and Blue box denotes nouns in a sentence.

For each of the floorplan variable O_1 to O_n, a class label is assigned to each floorplan variable. The probability distribution of noun *I* for which the floorplan variable that was created gives the score(w_{ij}) of the label *l* assigned to the variable. We create a dictionary of words which stores the meanings of words in mathematical form. For nouns, this information is represented in dictionary of words as a discrete probability distribution over class labels.

Now, for the presentation of the prepositions and prepositional phrases, we use location (denoted by μ) and concentration (denoted by κ) parameters for two independent Von Mises distributions [5]. The Von Mises distribution for angular distribution α is given as:

$$v(\mu,\kappa) = \frac{e^{\kappa cos(\alpha-\mu)}}{2\pi I_0(\kappa)} \quad (1)$$

When the i^{th} preposition in the lexicon is applied between two variables, whose physical relationship is specified by the position angle θ and velocity angle γ between them, its score z_i is given by

$$z_i(\theta,\gamma) = \left(\frac{e^{\kappa_{i,1}cos(\theta-\mu_{i,1})}}{2\pi I_0(\kappa_{i,1})}\right)\left(\frac{e^{\kappa_{i,2}cos(\theta-\mu_{i,2})}}{2\pi I_0(\kappa_{i,2})}\right) \quad (2)$$

Conditional probability of graphical model will be calculated using path variable $p\left(p_x, p_y, p_{v_x}, p_{v_y}\right)$ of the model on the K floor plan objects ($O_1 \ldots O_k$) and m assignments of labels to N floor plan variables. In mapping m_n, every element of mapping is the index of floorplan object mapped to floorplan variable n. This helps in deciding which objects in floor plan represent which noun in sentence.

Let α be $\{p, o_{m_1}, \ldots, o_{m_n}\}$ a set of path variable and mappings of floor plan objects to variables and b_{c_1}, b_{c_2} be indices of target and referent in *a.* Position of target is found by coordinates $\left(a^x_{bc_1} a^y_{bc_1}\right)$ and referent by $\left(a^x_{bc_2} a^y_{bc_2}\right)$. Similarly, velocity of target can be found by $\left(a^x_{bc_2} a^y_{bc_2}\right)$. Hence, position angle θ and velocity angle γ is

$$\theta = tan^{-1}\left(\frac{a^y_{bc_1} - a^y_{bc_2}}{a^x_{bc_1} - a^x_{bc_2}}\right)$$

$$\gamma = tan^{-1}\left(\frac{v_y}{v_x}\right) - tan^{-1}\left(\frac{a^y_{bc_2} - a^y_{bc_1}}{a^x_{bc_2} - a^x_{bc_1}}\right)$$

Now we calculate the segment conditional probability given mapping *m*, floorplan *f*, and dictionary Λ using the formula

$$\psi(p,m,f,\Lambda) = \prod_{c=1}^{C} z_{dc}\left(\theta_c,\gamma_c\right) \prod_{n=1}^{N} w_{e_n l_{mn}} \quad (3)$$

3.2 Training HMMs

Now, for each path-sentence pair in training data, we generate Hidden Markov Model (HMM). The states of each such HMM consists of probabilistic graphical model of each segment of sentence in proper alignment. Thus, number of states in HMM is equal to number of temporal segments. The HMMs infers the alignment between the densely sampled points in each path and the sequence of temporal segments in its corresponding sentence. The Output model for temporal segment t is given by

$$R_t(p_t,f,\Lambda) = \sum_m \psi_t\left(p_t|m,f,\Lambda\right) \quad (4)$$

Each HMM has its own transition and emission matrices (a and b respectively). Transition matrix in this problem is only allowed to self-loop or transition to next temporal segment state and must start at first segment and ends at last. HMM computes a score at every state to measure the degree of truthfulness of segment. The Expectation Maximization [6] Algorithm helps to find Log Likelihood of HMM being in each state. Once we have the likelihood and the transition model, we use it in Baum-Welch (Forward-Backward) Algorithm which is given as Algorithm 1.

Before learning the word meanings, all preposition and noun distributions are randomly initialized. During learning, the model is updated iteratively to increase the overall HMM likelihood as a product over all training

samples. At each iteration, this concentrates the Probability Mass Distribution of each HMM state's preposition distributions at angles seen at regions of the path during which that state is of high probability. It also concentrates the probability mass of the object label distributions in containers associated with the mappings corresponding to high HMM likelihoods. Thus, the parameters of nouns and prepositions in the dictionary is updated accordingly. Fig-3 illustrates how HMM is created.

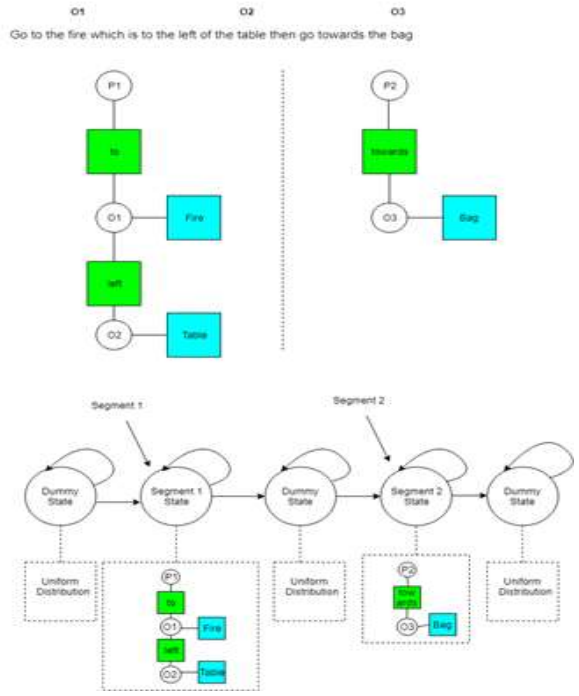

Fig-3 Creating HMM from a graphical model.

Algorithm 1 Baum-Welch Algorithm

Randomly Initialize distributions (θ)
repeat
  Compute forward messages: $\forall_{i,t}\ \alpha_i(t)$
  Compute backward messages: $\forall_{i,t}\ \beta_i(t)$
  Compute posteriors:

$$p(z_t = i \mid \mathbf{x}, \theta) \propto \alpha_i(t)\beta_i(t)$$
$$p(z_t = i, z_{t+1} = j \mid \mathbf{x}, \theta) \propto \alpha_i(t)p(z_{t+1}=j|z_t=i) \times \beta_j(t+1)p(x_{t+1}|z_{t+1}=j)$$

  Update θ
until Converged

Algorithm 1: Baum-Welch (Forward-Backward) Algorithm

Algorithm 2 Rule-Based Sentence Classification

```
procedure SENTENCEPARSER(sentence)
    doc ← sentence
    For token in doc:
     loop:
          Split over the verb not following determiner OR Split over adverb
          Remove segments which are not proper verb clauses
    For segment in doc
     loop:
          nouns ← Get nouns from sentence
          prep ← Get prepositions from sentence
  For preposition in prep:
  loop:
   if token.parent.parent is noun then
       subject ← noun.
   if token.parent.parent is verb then
       subject ← verb.parent.
   if token.parent.parent is preposition then
       subject ← token.parent.parent.leftchild.
   if token.child is noun then
       object ← token.child.
   if token.child.child is preposition then
       object ← token.child.child.leftchild.
```

Algorithm 2: Rule based Classification

4. COGNIZANCE AND PATHFINDING

We will now use a set of waypoints generated from sequence modelling for generating path for robot locomotion. This phase determines the sparse sequence of waypoints which satisfy the sentence.

4.1 Gradient Ascent

Gradient ascent algorithm is used to maximize the likelihood function. By maximizing, we try to find out the local maximum for the function. The likelihood function we try to maximize is Equation (4). We apply gradient ascent on score function of each temporal segment it to obtain small set of waypoints for each temporal segment. Thus, for every sentence we get sparse set of waypoints that satisfy the semantics of sentence.

4.2 Collision Awareness

The sparse set of waypoints which are generated by gradient ascent does not consider the fact that by following the waypoints, the robot can collide into an obstacle or other floorplan object. This is very important to be considered as obstacle avoidance should be a task that has to be done by the robot to function in real-world environments. Hence, we introduce barrier radius around each object in floorplan. An alternative to this is Barrier Penalty as suggested in [1]. A barrier penalty B(r) is added between each pair of a waypoint and floorplan object as well as between pairs of temporally adjacent waypoints to prevent them from becoming too close.

$$B(r) = SMOOTHMAX\left(1{,}1 + \frac{2r_1 + 2r_2}{r}\right) \quad (5)$$

Where r is the distance either a waypoint and an object or between two waypoints and where r_1 and r_2 are the radii of the two things being kept apart, either the robot or an object. This barrier is approximately 1 until the distance between the two waypoints become small, at which point it decreases rapidly, pushing them away from each other by approximately the robot radius.

4.3 Path Finding

Path finding algorithms are the algorithms that find the least cost route to target from given position in the environment. Thus, optimized path is generated if start and end points are given. Some of the commonly used Path-Finding Algorithm is A* Algorithm. The sparse set of waypoints is passed to path finding algorithm which generate an optimized path between initial position to target position.

Unity provides us with the tool to create a Navigation Mesh on the floor or terrain of the environment. NavMesh is a data structure which helps robotic agents for pathfinding through complicated areas, turns etc. It defines what part of environment is traversable and non-traversable. Using NavMesh we can set the target and the waypoints obtained from gradient ascent as the points through which robot agent should definitely pass through. The NavMesh Implementation is shown in Fig-4. Every Floorplan object is considered an obstacle with a fixed radius around it not a part of NavMesh hence creating barrier for collisions.

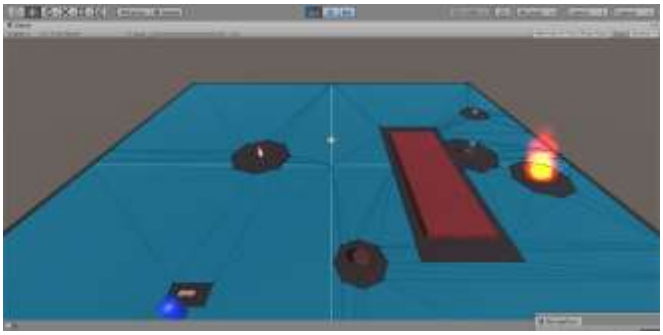

Fig 2.2 NavMesh in Unity Environment (Blue) and barrier radii for objects (Black)

Thus, Path is created from sparse set of waypoints and robot is able to understand the sentence and move accordingly in the environment.

5. FUTURE WORK

The framework we have discussed certainly is not complete in itself. The framework deals only with nouns and prepositions in the sentences but fails to recognize the role of verb in the sentence. This framework can be made generalizable framework for any robotic task by natural language if we are able to model and learn meanings of verbs, adverbs etc. in context of robot. If robot can understand what the verb in the sentence means then any task can be understood and done by same robot. Similarly, in Sequence Modelling, we could have used Temporal Convolutional Networks (TCN) [7] as it is better than Recurrent Neural Network with LSTM and can replace HMM for sequence modelling task. Given a sentence as an input, this would generate corresponding waypoints for each segment. Adding speech recognition engine would make this system more user-friendly and connected. However, these tasks are not currently in the scope of the paper.

6. CONCLUSION

We have understood how the framework would enable the robot to do tasks based on natural language understanding. Data collection is done for collecting raw data followed by preprocessing on the data. The robot will try to identify the words in sentences and find out nouns and prepositions, following which it is grounded by use of mathematical probability distributions and Hidden Markov Model for sequence. Now optimization by gradient ascent for the likelihood function is done for each temporal segment to arrive at sparse set of waypoints. These sparse set of waypoints are then given for calculating path through all these waypoints without colliding with obstacles and other objects. Hence the robot first learns the dictionary of word meanings and then use it to understand the natural language sentence and move accordingly.

7. REFERENCES

[1] Paul Barrett, Daniel & Alan Bronikowski, Scott & Yu, Haonan & Siskind, Jeffrey. (2017). Driving Under the Influence (of Language). IEEE Transactions on Neural Networks and Learning Systems. PP. 1-16. 10.1109/TNNLS.2017.2693278.
[2] Unity Game Engine: https://unity3d.com/unity
[3] "Industrial-Strength Natural Language Processing": https://spacy.io/
[4] Koller, Daphne and Nir Friedman. "Probabilistic Graphical Models - Principles and Techniques." (2009).
[5] M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions. New York, NY, USA: Dover, 1972
[6] Dempster,A.P. ; Liard,N.M.; Rubin,D.B.(1977) Maximum Likelihood from Incomplete Data via the EM Algorithm.
[7] Bai, Shaojie & Zico Kolter, J & Koltun, Vladlen. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modelling.