Demo Paper:

From Virtual to Real: A Framework for Verbal Interaction with Robots

Eugene Joseph,
North Side Inc.
{*eugene-at-northsideinc-dot-com*}

Abstract

A Natural Language Understanding (NLU) pipeline integrated with a 3D physics-based scene is a flexible way to develop and test language-based human-robot interaction, by virtualizing people, robot hardware and the target 3D environment. Here, interaction means both controlling robots using language and conversing with them about the user's physical environment and her daily life. Such a virtual development framework was initially developed for the Bot Colony videogame launched on Steam in June 2014, and has been undergoing improvements since.

The framework is focused of developing intuitive verbal interaction with various types of robots. Key robot functions (robot vision and object recognition, path planning and obstacle avoidance, task planning and constraints, grabbing and inverse kinematics), the human participants in the interaction, and the impact of gravity and other forces on the environment are all simulated using commercial 3D tools. The framework can be used as a robotics testbed: the results of our simulations can be compared with the output of algorithms in real robots, to validate such algorithms.

A novelty of our framework is support for social interaction with robots - enabling robots to converse about people and objects in the user's environment, as well as learning about human needs and everyday life topics from their owner.

1 Background and motivation

If robots are to suitably collaborate with humans in tasks commonly occurring in settings like the home or at work, natural language interaction with robots is a must. Connell (2018, 2012) argues that the most efficient interface to universal helper robots, at least for the elderly, is direct speech.

North Side Inc. (www.northsideinc.com) originally developed a Natural Language Understanding (NLU) and Generation (NLG) pipeline (Joseph 2012, 2014) for its Bot Colony video game (www.botcolony.com). The game is based on the Bot Colony science-fiction novel (Joseph 2010), which anticipated the functionality of verbal interfaces of intelligent robots circa 2021.

An all-graphics (virtual) framework removes the constraints related to real hardware, enabling one to focus on refining an intuitive language-based human-robot interaction. With hardware out of the way, it is easier, faster and cheaper to make progress, and virtual robots are acceptable interlocutors (Bainbrige, 2008). All the functions described in the paper were implemented and can be observed in Bot Colony.

2 Requirements for natural verbal interactions with robots

When interacting verbally, a person would expect a robot to understand whatever he/she says just as well as a member of our species would (Bisk, 2016). Capabilities (i) – (v) below, implemented in our framework, are innovative features of the framework.

(i) To link language to actions and objects in the real world, one should be able to refer to objects or people using natural speech – using similar words, similar syntax, and using pronouns, proper names and determiners to refer to entities. (ii) The spatial language understood by a robot should be full English (or another natural language; our framework currently works only in English, but high quality translation to other languages is now available and could be integrated). (iii) For natural interaction, the major Dialog Acts used in human conversation (question, fact statement, command, opinion, Yes and No answers, etc.) should be supported. (Stolcke, 2000) (iv) The conversation should be multi-turn and (v) Interlocutors should have the ability to refer to context.

These capabilities represent advances over work such as Connell (2018) and others (listed in Bisk, 2016) where structured languages with small vocabularies and grammars specify the acceptable syntax, spanning only a small subset of full English. The rest of the paper describes key aspects of our implementation.

3 Grounding Language References to Entities

People refer to actions and entities in many different ways, using their own words. Resolving references to individual entities is a major problem in NLU, known as coreference resolution. See Elango (2006) for a survey of the domain. In particular, resolving a reference should result in a robot knowing the current position of the referred entity, so the robot can manipulate it. While object recognition is required to manipulate an object, it is clearly not sufficient: a robotics application in a large warehouse will need to process references to many thousands of objects. In a household, a robot owner will refer to people and hundreds (or even thousands) of objects. An Entity database, an innovative feature of our framework described in section 4, can resolve referring expressions in a larger applications. The Entity Database is distinct from databases containing object models used in object recognition tasks, like the ROS Household Object database (ROS.org).

A key challenge is referring to *instances* of objects - one of several individuals of the same type. When a robot is unable to resolve a particular instance (*Pick up the guitar* in a room with 3 guitars), it will ask questions like *Which guitar? The blue guitar, the black guitar, or the silver guitar?* Clicking on an instance is one way to resolve the instance and is an example of the coordination Clark (1996) referred to. However, this clicking to disambiguate may not translate well to a real application. Our framework is able to resolve object references using language, the way a person distinguishes objects of the same type: by specifying an attribute of the object, a relation to another object, an index in a list offered by the robot, an object state, or by elimination. For example:

- *the blue guitar* (color attribute)
- *the guitar on top of the bed* (spatial relation to another object)
- *the first one/ the last one* (index in a list, resolving respectively to the blue guitar or the silver guitar in the example above)
- for objects that have states, the state of an objects can be used to specify the instance desired (*the open one* – for something like a drawer or a door).
- When there are two objects of similar type, the robot will point to one of them and ask *This one? Say Yes or No*. (discrimination by elimination)

In later versions, the robot defaults to the instance closest to it, to reduce the need to clarify the instance. If a robot makes an undesirable choice, the user can say something like *go to the other one*, and a robot would move to the next instance which is spatially closest.

The examples above deal with distinguishing individuals of the same type. There are other cases requiring resolution of linguistic references (using the user's words) to entities:

- anaphoric (pronominal) references (*pick up the green briefcase, put <u>it</u> on the scanner*; 'it' refers to the green briefcase),
- *you* (the interlocutor)
- *they, them* (intelligent agents vs objects)
- a child concept from the taxonomic parent, as in *pick up the toy* (the toy giraffe, if it's the only toy in the environment),
- *here* (reference to a place in remote control situations),
- *there* (to a previously mentioned place)
- temporal time-point resolution, like *then, next, first, last* (see Perceptual Memory)

All these resolutions are supported by the Coreference Resolution module of our framework (see diagram in 8). In general, coreference resolution differentiates between ground entities (in EDB) and non-ground entities appearing in discourse (eg *I like horses*). The coreference algorithms rely on the EBD and the ontology described next.

4 The Entity Database

A key part of the solution to general grounding and coreference resolution is the Entity Database (EDB) – a database containing information about all the entities (any physical object, location) in the environment (Tellex (2011) refers to a location map for grounding). The EBD is required for the critical co-reference resolution task described above.

For a virtual simulation, the EDB can be created with tools like 3DSMax (which can import Autocad). The EDB can be exported to Excel, edited there and re-imported into a commercial Object-Oriented database used to store the Entity Database (Versant Object Database, Actian Corp). The challenge is building a EDB for the real application.

In a real application, object recognition is necessary. Databases like ROS household_objects SQL database (ROS.org), could be used in the object recognition task, which, together with the ROS database, would contribute the information needed to generate an EDB as described below, for coreference resolution purposes.

Finally, for industrial application, CAD models used in manufacturing could be used to train ML for object recognition. Majumdar (1997) explores using CAD models for object recognition.

An innovation in our framework is an extensive ontology containing **all** English nouns, built from MRD's (machine-readable dictionary, such as Wordnet). Through the ontology, knowing the type of an object (disambiguated to its sense number in the MRD) gives one access to its ontological parent, its parts, its attributes, its purpose, etc. This is a major improvement over Connell's approach which does not use an ontology. Connell's 'teaching approach' is bound to introduce problems, as humans cannot be expected to use language formally (in Connell (2018) the users are elderly people). Take Connell's example to teach a 'supporting shelf' concept: 'Supporting shelf – the LOCATION is a supporting shelf'. Ontologically, a supporting shelf is not a location, it is **in** a location. Artifacts (such as shelf) have very different properties from fixed locations. If formal reasoning were used on learnt concepts, imprecisions in definitions could have undesirable effects on robot task success (eg, if locations named differently should be different, it could be difficult for a lower shelf to be at the location of an upper shelf).

Irrespective of how the EDB is created, every object that needs to be referenced through language must be in the EDB. Objects are given common English names (eg, chair), and alternate denominations are supported, to support human references naturally. When several objects of the same type are in the same space, an instance number is appended to the type of the object to form its name. While the instance number is currently entered manually, assigning instance numbers to objects of the same type at different coordinates could be automated. In addition to a type, objects in EDB have geometric properties and parenting (on top of) scene information. In the virtual framework, attributes are given values with a tool like 3DSMax. In the real application they would be set through object recognition, and scene information. The attributes required by the virtual framework include X, Y and Z coordinates, dimensions, colors and textures, the parent (the object on top of which another object is), and a 3D position and orientation of the bounding box of the object relative to the origin of the coordinate system.

5 Spatial Relations

For natural verbal interaction, users should be able to refer to spatial relations the same way they do in everyday life. In our virtual framework, these relations are not difficult to compute, as we have the coordinates of every object in the environment, and their bounding boxes. In the virtual environment, computing a spatial relation nvolves comparing the coordinates of the relevant planes of the bounding boxes of the participating objects. This approach could be emulated in the real application, provided object recognition works well enough. Examples of spatial relations computed in this way are: X in the center of Y, X in front of Y, X to the left/right of Y, X under Y, X on Y, X behind Y, X in front of Y. An important design consideration for spatial relations is computing relations like left of Y/right of Y, in front of Y/behind Y from the point of view of the human user of the robot – as an object Y to the left of the robot will be to right of a user/operator facing that robot (or looking through a camera that faces the robot). In our framework, these relations are computed relative to the human user's camera.

At any point, a robot knows (and can tell when asked) the distance between any two objects (so in particular, the distance from itself to another object). The angle from the center of the robot's viewing frustum and another object is also computed and available in the database of observations that ground the robot event memory.

6 Knowledge Grounding and the Robot's Perceptual Memory

To interact with robots efficiently, a user needs to be able to find out what the robot knows. This comprises the commands it understands, the tasks it can execute, its 3D environment, its perception of events in this environment, and its background knowledge. In our environment, *What do you know?* is the first step in exploring a robot's knowledge. The wording of the robot's answer is intended to stimulate further interactive exploration of the knowledge (which can be vast) - by drilling down with additional questions.

Grounding means grounding basic language phrases to perceptual or motor skills, objects and locations. The grounding of objects and locations (the 3D environment) in our framework was described above. Facts known to a robot and observations made are labelled with a source of knowledge. The **sources of knowledge** are A) perception (SEE, HEAR events) – the perceptual memory is described in detail below, B) communication with other intelligent agents (a blackboard of sorts) and C) factory (static) knowledge. Commands are mapped to motor skills (see Commands).

Any robot in our framework will answer that it knows:

(i) its environment
(ii) its commands
(iii) its job
(iv) facts it was taught
(v) events it witnessed
(vi) general concepts.

These can be expanded, eg with facts about named-entities such as supported by, say, Alexa or Google (sports teams, artists, bands, movies, etc.) and world knowledge (see Future Work in 9). Categories i) – vi) are explored below.

6.1 Environment knowledge

Implementing the EDB concept in the real application will provide 'out of the box' support for the verbal interactions described below:

Spatial relations can be used in questions *What's on the table?* , or in commands *Put the vase between the candles.*

Scene contents *What do you see?* can be useful to test the vision and object recognition capabilities of remotely-controlled robots. A robot will answer a "What do you see?" question with a description of the objects in its viewing frustum.

Our framework can simulate **Robot control in remote settings**. Mediated interface to a robot can be via virtual devices such as a tablet, or cameras (in our framework, ceiling or wall cameras for interior spaces, or exterior cameras installed, for example, on an oil rig).

Number of objects in a container or area *How many cups are in the cabinet?* As certain questions like *What's in this room?* can return lengthy answers, any robot obeys *Stop* – which interrupts execution of the last command.

Questions about the **attributes** of an object in the environment *What's the height of the fridge? What's the color of the vase?* are supported directly using the EDB.

Distances in the environment *What's the distance between X and Y?* is supported using object coordinates in EDB.

Information on an object in the environment In the virtual environment, the user clicks on an object, asks *What is this?* and the robot answers with the type of the object from EDB. In a real application, the user would click on a point in the image returned by the robot's vision sensor, and the recognized object type would be used to query the EBD, using knowledge of the robot's current position and position of each object in the scene – to identify the particular instance of that object type.

6.2 Robot Commands

A robot's command set is explored with *What are your commands?* Depending on the robot, this may return *I know some movement commands, some manipulation commands, and some communication commands*. Asking *What are your* (category) *commands*? produces a list of the commands in that category.

Movement commands

- *Go to* <place> (Go to the bedroom). The robot will move to <place>.
- *Go to* <object> (go to the vase) the robot will turn to face <object>.
- *Face* <object> If not already facing <object>, the robot turns to face it.
- *Turn clockwise/counterclockwise* (*by Y degrees*)
- *Move forward/back (by Y meters)*
- *Stop* (to reset a robot).
- *Follow me, stop following me*. This command is useful in a videogame played in 3rd person, but could be changed to *follow X* (another robot).
- *go up, go down* (a robot moving on a rail is able to translate up/down or extend the manipulator arm to grab baggage from a shelf).
- *Jump* (unlikely command in a real application!)

Manipulation commands

- *Pick up* <object> (Pick up the vase). Implemented as face, reach and grab, see below. The user can ask "What do you hold?".
- *Grab* <object> (part of pick up X)
- *Drop* <object>
- *Push in* <object> (push in the cushion). *Close the drawer* works as an alternative to push in.
- *Put* <object1> *on* <object2> (put the red box on the blue box). *Put* object1 *to the left/right of* object2 (space availability is checked).
- *Put* <object1> *between* <objects> (put the vase between the candles, put the bottle between the sinks).
- *Put* <object1> *in the center of* <object2>. Knowing all dimensions enables us to check space availability prior to execution in a simpler way than in Howard (2014).
- *Rotate* <object> *by Z degrees clockwise/counterclockwise*
- *Swap* <object1> *with* <object2>. *Put* <object1> *where* <object2> was - also works.
- *Align* <object1> *with* <object2> (the user needs to imagine that he/she is on a plane or ship looking FORWARD, seeing a red light on his left and a green one on his right. The left (red) and right (green) and an arrow showing the forward direction of the reference object are superimposed on the reference object, and a yellow arrow is attached to the target object. The framework asks *Where should the yellow arrow point?* and differentiates two cases: when the target object is on top of the reference object, or when the two objects side by side.
- *Open door (open cupboard door* - in the kitchen); *Close the door (or the drawer, the guitar case)*

Body-part commands

- *Reach for <object> (part of pick up X)*
- *Point to <object> (or point to room)*
- *Wave*
- *Nod*

Expressing commands in different ways can be currently done with the Command Teaching facility (below). In the future, synonymic commands will be supported with semantic frames (see Future Work in 9).

Command Execution

Validation When a robot cannot execute a command (because an object is not reachable, is not movable, it is too large/small, there's not sufficient space to place an object, or because the robot is already at the destination) it will provide a diagnostic. If a command missed an argument, the robot will query for the missing argument (*go where?*).

Help A Help function is available. For complex commands, visual guidance and interactive help are available as described above for align. In Jimmy's World (see Future Work in 9) help is available conversationally.

Execution and Grounding to Motor Skills In our implementation, a robot first navigates towards the target and then turns to face it. Collision avoidance in the virtual environment is done with Havok AI, which supports 3D path planning. Collision avoidance in real applications requires sensing obstacles and avoiding them, and our movement commands could support this if necessary.

The robot moves close to the target using forward kinematics. If required, a humanoid will bend at the hips and knees while its effector starts reaching forward. This position becomes the starting position for inverse-kinematics (IK) movement of the robot effector. A similar approach that tracks state changes of objects during manipulation and after it was described in Zielinski (2015). Our framework uses HumanIK for inverse kinematics. Optimal grabbing of objects is an important area in robotic frameworks. In our framework, collision detection with Havok Physics ensures that the robot's manipulator does not go through the object is manipulates. We've implemented finger placement algorithms that rely on automatically generated 'grabbing points' (placed on opposite faces of small objects, or towards the end of larger objects) so that grabbing objects looks natural. Grabbing points can also be edited by users. Our framework supports both one-handed and two-handed grabbing of objects.

Movement to a target point, body/head rotation, effector rotation, reach and grab are basic motor skills supported in a server-side client script engine (CSE), to ground the higher level English commands listed under COMMANDS. Translating English to atomic robot commands is demonstrated in the Jan 2013 video (A.1). This approach was described as early as 2001 in Nicolescu, is used in Kress-Gazit (2008), Matuszek (2013) and in Misra (2014) [which uses pronouns without mentioning coreference resolution].

In our framework, objects have physics implemented with the commercially available Havok Physics tool (so an object falls if a robot drops it).

Teaching New Commands Required to Streamline Tasks

A robot will offer to learn a new command if it doesn't know it. Commands are entered one by one, and at execution time, they will adapt to new target objects. The initial version of Bot Colony launched in June 2014 supported learning new commands described as a sequence of existing (native) commands, where objects are parameterized. A similar approach was described subsequently in (Gemingnani, 2015).

EXAMPLE *scan the green briefc*ase The robot replies that it doesn't know 'scan', and ask if the user wants to teach it. Commands are entered one by one: *go to the shelf, pick up the green briefcase, go to the scanner, put the briefcase on the scanner, End.*

Co-reference resolution kicks in during execution to resolve the particular shelf (e.g., upper Tokyo shelf).

6.3 Robot tasks

A robot should be able to tell a user about the higher level tasks it can accomplish. Our framework treats a task like a new command, built from individual commands. Since our application was a videogame, there was no need to ground robot tasks to skills and objects, and these cannot be demonstrated in the framework. The conversation related to tasks was prototyped for use in the videogame and looks like this:

What is your job? I can clean the house, cook, wash dishes, do the laundry, babysit,…How do you clean? I vacuum the floor, I dust the furniture, I mop, etc.

However, if the user asks the robot to mop the floor, he'll learn that this function is not currently working.

6.4 Factual knowledge

For home or companion applications, knowledge of the owner and his family would enable a robot to resolve references and understand the context. Our framework supports configuring a robot with the knowledge required to serve a particular owner and family by reading in a fact base and updating EDB (it is also possible to give facts conversationally at run time). The Question Answering (QA) component can be used by a user to explore a robot's knowledge.

EXAMPLES *Who are the members of the family?Who are Ayame's children? Who is Hideki? When does X usually come home from school? What do you know about X? (Hideki is 8 years old. Hideki is the son of Ayame. Hideki goes to school). What games does Hideki enjoy? Is Masaya married? Where does Masaya work? What does the family eat for breakfast? How do you prepare X?*

Technically, these questions are not more difficult to answer than the ones Alexa or Google Assistant answer on named-entities like cities, restaurants. Conversely, if the necessary information were available, the QA component of our framework would enable a robot to fulfill functions of smart speakers, in addition to performing its physical tasks.

6.5 Perceptual Memory and Grounding of Robot Perceptions

An innovative feature of our framework is logging a robot's salient observations – events the robot witnessed- and making these accessible through question answering (QA). This is important, for example, in a security application (*When did the XYZ truck come in? When did it leave?*).

In our framework, salient observations are

- objects of interest (people, vehicle, animals – any type declared as being of interest, or OOI) entering or exiting the robot's field of view. An OOI entering/exiting the field of view triggers logging the sighting (or the speech, if applicable) for the particular type of intelligent agent or object.
- a person performing an action
- a person speaking
- any action performed by the robot

Visual and audio observations are time stamped (YYYY-MM-DD HH:MM:SS) and have a range and angle to the target. "*M. arrived on 19 August 2021 at 01:10 AM*". "*How do you know*?" "*I've seen M. from 7.2 m at an angle of 40 degrees*". In our framework, salient observations of a robot can be played back (since we control all the actors, they are actually re-enacted on the fly).

Assigning semantics to observed actions like in "M. hid the chip in the toilet water tank" is easily done in a simulated environment, but is more challenging in a real application (how can a robot tell that someone is 'hiding' a chip?). In a real implementation, a robot could be able to recognize people and objects, and some basic actions and states of people (moving near an object, interacting with objects, sitting, lying down, coming into view, becoming not visible). Connell is proposing solution for gesture recognition in Connell (2018), but it's not clear if these can be extended to recognizing actions.
Here are some of the most useful questions supported for exploring grounding (note temporal resolution of 'first/last' 'then', 'next', 'before that', 'after that'):

- *What did X do at HH:MM on Day/Date*? (example: What *did Ayame do at 20:15 on Thursday?*) *What did she do **then**? What did Masaya do **next**? What did he do **before that**? How do you know that?* (grounding) *When did you first/last see X? What happened then? What happened before/after that? What happened at HH:MM on (day of week)*? (What happened at 11:30 on 26/08/2021? – this will work even if after/before don't return more facts because Jimmy the robot didn't look back/forward far enough. *When did X arrive/enter/leave the house? Where did X go after that? What did X say at (time) on (day)? What did X say before/after that? Where was X at (time) on (day of the week)/date**?***

6.6 Generic Concepts

The framework supports accessing a dictionary, useful to non-native speakers of English. Intelligent conversation going beyond a definition, about any concept, requires massive knowledge about the world. In our forthcoming Jimmy's World, Jimmy (or whatever the player names his embodied bot) is able to converse on any concept and learn from the user and the community. The objective is to understand how a concept fits into everyday life. (Joseph, 2019)

7 Framework Implementation

The architecture of our NLU pipeline is shown below. The pipeline software runs on a Linux server that communicates with client software using the Google/protobuf protocol. The client manages the 3D world and robot animation, and users can interact through speech or typing. The client implements English commands sent from the server using the ground motor abilities described above. Voice input is processed by the client which calls cloud-based speech-to-text, sending the resulting text to the server-based NLU pipeline. After the pipeline generates the response, text-to-speech server-side sends audio files to the client. Language-understanding is grounded as explained in this paper. A logging service logs all interactions, and Competency (non-IDK [I don't know] answers) - is reported as a percentage of all utterances (see below). Our virtual framework represents major types of robots operating in various real environments (see images to the right). Bot Colony prototype scenes include: a home, an airport with a baggage warehouse, an oil rig, a hotel, a village filled with robotic vendors, entertainers and waiters, a hotel with robotic personnel, a manufacturing facility, a military installation, a harbour and a mine. A variety of robots are supported: humanoid robots, fixed-base greeting robots, mobile observation robots (camera bots), a rail telescopic robot, military robots, flying robots (Hunter bot) – each with commands adapted to their tasks (see Commands section).

8 Comparison with Related Work

While small vocabularies and grammars are the norm (Bisk, 2016), our pipeline supports **full** English, including idioms and phrasal verbs, in conversation. Another major novelty in our pipeline (see diagram below) is using syntactic and semantic rules mined from dictionaries for higher precision. For example, our parsing component combines the Stanford Parser, Berkeley parser and our own Template Parser, which uses syntactic rules mined from dictionaries. This parser is used to parse robot commands with very high precision (in excess of 95% on well-formed commands). On other Dialog Acts, we achieve a precision slightly superior to the component Stanford and Berkeley parsers, as we've repairing systematic parsing errors made by these parsers.

As explained below, we are currently transitioning our disambiguation to semantic frames. Coreference resolution with EDB is designed to be interactive and seek user clarification when necessary – so precision is high for entities that are in EDB.

We are logging game sessions and we compute a Competency metric (%age of utterances that don't cause *I Don't Know* answers). As players often refer to unknown entities or facts – Competency can vary widely from session to session. However, on 400 longer sessions (above 300 dialogue turns) the average Competency observed was 69%.

Figure 1: Humanoid robot

Figure 2: Telescopic rail robot

Figure 3: Airborne hunter robot

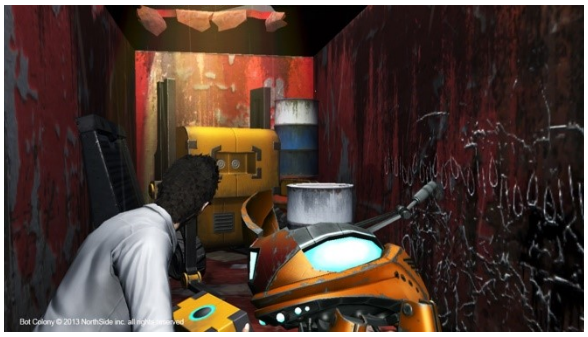

Figure 4: Underwater welding robot with welding torch and tools

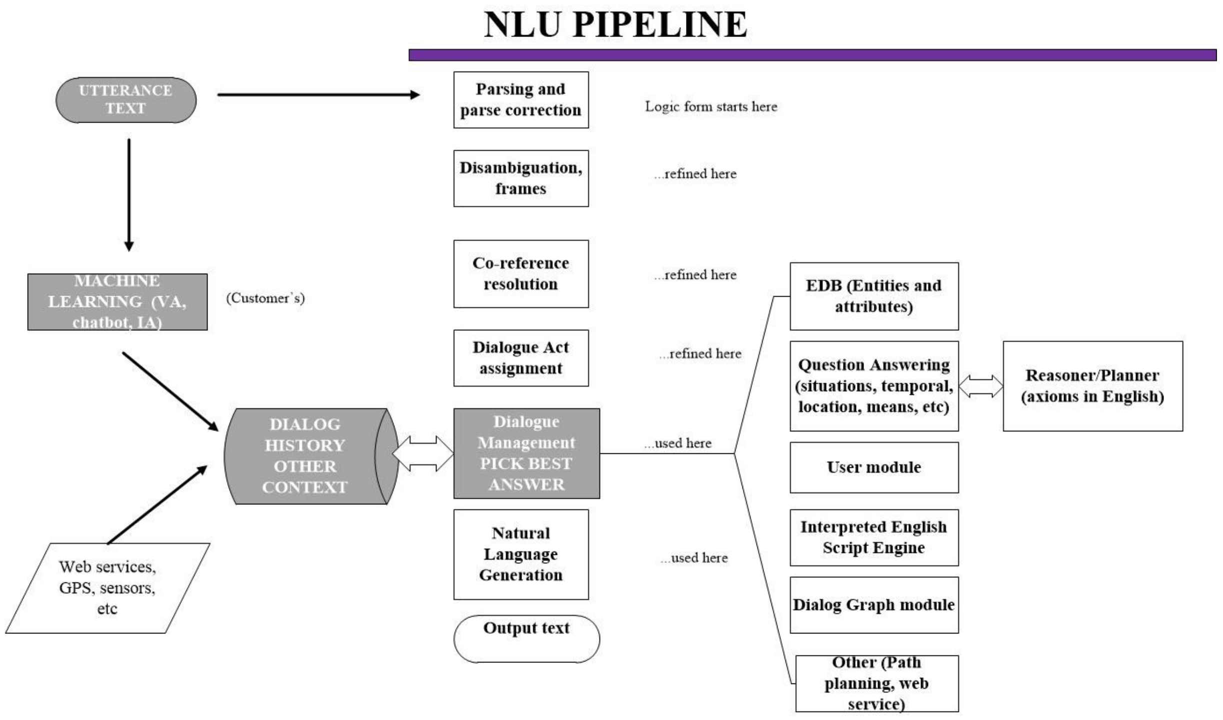
NLU PIPELINE
UTTERANCE TEXT
Parsing and parse correction
Logic form starts here
Disambiguation, frames
...refined here
Co-reference resolution
...refined here
Dialogue Act assignment
...refined here
MACHINE LEARNING (VA, chatbot, IA)
(Customer's)
DIALOG HISTORY OTHER CONTEXT
Dialogue Management PICK BEST ANSWER
...used here
Natural Language Generation
...used here
Output text
Web services, GPS, sensors, etc
EDB (Entities and attributes)
Question Answering (situations, temporal, location, means, etc)
Reasoner/Planner (axioms in English)
User module
Interpreted English Script Engine
Dialog Graph module
Other (Path planning, web service)

Figure 5: A key aspect of our NLU pipeline is a logic form that represents utterances formally. This logic form is initially produced from the parse tree of the input utterance. It is refined by the disambiguation module which adds sense numbers from the MRD sense inventory, and the coreference resolution module that grounds linguistic references to EDB entities. A reasoner applies axioms to this logic form, to infer, eg, that cats are born and die. Dialog Mgmt and Natural language generation also use this logic form.

9 Conclusion and Future Work

The next frontier is teaching a robot about everyday life and user preferences – a fusion between robots and intelligent assistants. This is the focus of our more recent work in Jimmy's World (Joseph, 2019). If the physical functions of a robot can be complemented with a robot ability to act as an intelligent assistant and companion - universal helper robots may become a compelling offering, especially for the elderly and people who live alone.

The Bot Colony architecture dealt with the basic issues of situated (3world based) NLU: coreference resolution, commanding robots, exploring a robot's event memory, etc. In Jimmy's World our focus is on knowledge-based NLU, so the acquisition and use of knowledge about everyday life in conversation – to cater more to personalized, intelligent assistant part of a robot's mission. A player's virtual robot in Jimmy's World will have curated knowledge from dictionaries, but will also learn from the user and the community.

Semantic frames are a way to understand language independent of the particular words and syntax used. Disambiguation to semantic frames, instead of the focus on individual Word Sense Disambiguation, is a key area of work.
A major milestone will be acquiring knowledge from individual users and the community and filtering reliable knowledge from unreliable knowledge, humour, witticism, etc.

To achieve this, we will need to refine knowledge-representation mechanisms for everyday life knowledge, and to use this knowledge in reasoning and conversation.
Since a lot of everyday life is about attaining goals and overcoming obstacles, reasoning and planning how to attain goals is another important area of work.

Machine Learning based NLU provides excellent coverage. Complementing a Machine Learning pipeline with knowledge-based NLU of the kind we are developing will result in higher precision, and deeper understanding of user utterances and is of strategic importance.

References

Yonathan Bisk, Deniz Yuret, Daniel Marcu. 2016. Natural Language Communication with Robots, NAACL.

Herbert Clark. 1995. Using Language, Google Books.

Jonathan Connell. 2018. Extensible Grounding of Speech for Robot Instruction.

J. Connell. E. Marcheret, S. Pankanti, M. Kudoh, and R. Nishiyama. 2012. *Proc. Artificial General Intelligence Conf.* (AGI-12), LNAI 7716, pp. 21-30, December 2012.

P. Elango. 2006. Coreference Resolution: A Survey, Technical Report, UW-Madison.

Gemignani, G., Bastianelli, E., Nardi, D. 2015. Teaching robots parametrized executable plans through spoken interaction. In: Proc. of AAMAS.

H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. 2008. Translating Structured English to Robot Controllers, Advanced Robotics, vol. 22, no. 12, pp. 1343–1359.

Jharna Majumdar. 1997. A CAD Model Based System for Object Recognition, Journal of Intelligent and Robotic Systems, Volume 18, Issue 4, April 1997.

Eugene Joseph. 2010. Bot Colony – A Novel Set in the Present and Near Future, North Side Inc.

Eugene Joseph. 2012. Bot Colony – a Video Game Featuring Intelligent Language-Based Interaction with the Characters, Eugene Joseph, North Side Inc., GAMNLP workshop, a special session at **JapTAL 2012** (the 8th International Conference on Natural Language Processing), Kanazawa, Japan.

Eugene Joseph. 2014. Natural Language Understanding and Text-to-Animation in Bot Colony, Gamasutra, **http://www.gamasutra.com/blogs/EugeneJoseph/20140626/219765/Natural_Language_Understanding_and_TexttoAnimation_in_Bot_Colony.php**

Eugene Joseph. 2019. Jimmy's World: Making Sense of Everyday Life References, Conversational Interaction Conference, San Jose, CA, March 11, 2019. https://docs.wixstatic.com/ugd/dbc594_ef32fd33d5b94bd9b6038f5e524c89ba.pdf

C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. 2013. Learning to Parse Natural Language Commands to a Robot Control System. In Experimental Robotics, pages 403–415. Springer.

M. Nicolescu and M. J. Mataric. 2001. Learning and interacting in human-robot domains. IEEE Transactions on Systems, Man, and Cybernetics, Part B,special issue on Socially Intelligent Agents - The Human in the Loop, 31(5):419–430.

Dipendra K Misra, Jaeyong Sung, Kevin Lee and Ashutosh Saxena. 2014. "Tell Me Dave: Context-Sensitive Grounding of Natural Language to Manipulation Instructions", Robotics: Science and Systems.

ROS.org The household_objects SQL database http://wiki.ros.org/household%20objects .

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L Petruck, Christopher R. Johnson, Collin F. Baker, Jan Scheffczyk. 2016. FrameNet II: Extended Theory and Practice (Revised November 1, 2016).

Andreas Stolcke, Klaus Ries et al. Dialog Act. 2000. Modeling for Automatic Tagging and Recognition of Conversational Speech Computational Linguistics, Volume 26 Number 3, ACL.

S. Tellex, T. Kollar, S. Dickerson, M.R Walter, A.G. Banerjee, S. Teller, and N. Roy. 2011. Approaching the symbol grounding problem with probabilistic graphical models. AI magazine, 32(4):64–76.

Thomas M. Howard, S. Tellex, N. Roy. 2014. A Natural Language Planner Interface for Mobile Manipulators, ICRA 2014.

Cezary Zielinski, Tomasz Komuta. 2015. An Object-Based Robot Ontology, Advances in

Intelligent Systems and Computing, January 2015.

A Appendices

A.1 Bot Colony Tech Demo Jan 2013

https://www.youtube.com/watch?v=54HpAmzaIbs

A.2 Robot Perceptual Memory Video

https://www.youtube.com/watch?v=8zV1r8VxWRM