Language Modeling for Robots-Human Interaction

Lesia Kaigorodova[1(✉)], K. Rusetski[2], Kiryl Nikalaenka[1], Yuras Hetsevich[1], S. Gerasuto[1], R. Prakapovich[1], U. Sychou[1], and S. Lysy[1]

[1] United Institute of Informatics Problems, Minsk, Belarus
lesia.piatrouskaya@gmail.com, anak247@gmail.com, yury.hetsevich@gmail.com, stanislau.lysy@gmail.com, {contacts, rprakapovich, vsychyov}@robotics.by
[2] Belarusian State University of Informatics and Radioelectronics, Minsk, Belarus
rusetski.k@gmail.com

Abstract. Building systems for speech communication between robots and humans is considered to be a difficult task since nowadays speech recognition systems still have poor performance and at the same time the ambiguity of recognized commands is unacceptable for safety reasons. We introduce the SAOF concepts for language models to define standards for communication.

The article covers the approach for language modeling for robots-human interaction, i.e. a standard for robots' behavior in order to make them perform commands. We consider two tools that allow us to go through this process. The first one is the NooJ tool and the second one is the Open Semantic Technology for Intelligent Systems (OSTIS) tool. The NooJ tool is easy to apply and solve many NLP problems while the OSTIS claims to be a universal solution for the tasks dealing with semantics though quite tricky to use.

Keywords: Human-robot interaction · OSTIS · Language model · Syntactic grammar · SAO format · SAOF format

1 Introduction

Building the architecture of systems for controlling a robot or a group of robots through speech is a tricky and special task. It is much different from the architectures of the systems with manual control of robots. There are some substantial reasons for this statement. The two main issues are: poor speech recognition performance of speech recognition systems nowadays and the problem of safety while dealing with robots.

Today speech recognition software in general is still producing poor quality results. Especially this concerns languages that are not world-widely used. For instance, Slavic speech is still hard to recognize. All the problems that are connected with speech recognition systems are usually narrowed down to some simpler problems or problems with a set of restrictions in order to achieve more accurate results.

© Springer International Publishing Switzerland 2016
T. Okrut et al. (Eds.): NooJ 2015, CCIS 607, pp. 162–171, 2016.
DOI: 10.1007/978-3-319-42471-2_14

The problem of safety while dealing with robots can be resolved by obtaining higher accuracy and disambiguation of recognized commands. If we want to control and interact with a robot or a group of robots easily and safely, the communication problem should be solved by designing the language that is common to everyday use of humans and that is 'understood' by robots as easily and quickly as possible. It is easy for humans to understand speech, but it is not so easy to make machines do this.

Given that, we should define the appropriate standard for robot's behavior in order to make them perform commands. If we use speech as a means of communication, we can say that a language model in some way defines this standard, it defines robots behavior, i.e. what they can perform and what they cannot perform, what they can 'understand' and what they cannot 'understand'.

For language modeling we will use both NooJ and OSTIS tools. NooJ tool is easy to use and it allows solving many NLP problems. And as for OSTIS, it is claimed to be a universal solution for the tasks dealing with semantics but it is quite difficult to operate.

2 Concepts of the Language Model

We have already stated that we should define an appropriate language model to define the standard for communication between humans and robots. Now we will introduce some concepts for this model.

At the first stage of the design of the Language Model we use *deep syntactic analysis* to obtain the model that is as simple as possible and yet far from underfitting the real model. We will use such concepts as *'Subject'*, *'Action'*, *'Object'* and *'Features'* (SAOF format).

'Subject' refers to a robot's name. *'Action'* refers to an action to be performed by robots that is usually represented by a verb. *'Object'* represents a target of the action. And *'Features'* is an add-on to specify *'Object'* or *'Action'*.

3 Introduction to NooJ Tool

In our work we use *NooJ* tool for designing the Language Model for robots-human interaction. NooJ is a development environment mainly used to construct large-coverage formalized descriptions of natural languages, and apply them to a large corpora, in real time. It also can be used not only for pattern recognition in texts but also for the generation of some patterns.

Finite State Transducer is a great solution for generating languages or dictionaries for languages, especially when they are not too complex. This type of language representation is simple to implement, understand and modify. And we will use *Finite State Automata* for pattern location while performing speech recognition.

Here are some definitions of the concepts in NooJ terms.

A *finite-state transducer* (FST) is a graph that represents a set of text sequences and then associates each recognized sequence with some analysis result. The text sequences are described in the *input* part of the FST; the corresponding results are described in the *output* part of the FST.

In NooJ, *Finite-State Automata* (FSA) are a special case of finite-state transducers that do not produce any result (i.e. they have no output).

Typically, a syntactic FST represents word sequences, and then produces linguistic information (such as phrasal structure). A morphological FST represents sequences of letters that spell a word form, and then produces lexical information (such as a part of speech, a set of morphological, syntactic and semantic codes).

NooJ contains a graphical editor (as well as a dozen tools) to facilitate the construction of FSTs and their variants (FSA, *Recursive Transition Networks* and *Enhanced Recursive Transition Networks*).

NooJ users typically use FSA to locate morpho-syntactic patterns in corpora, and extract the matching sequences to build indices, concordances, etc.

4 Language Modeling Using NooJ Syntactic Grammar

In NooJ, *Syntactic Grammars* are the grammars that process sequences of tokens (as opposed to the morphological grammars that process sequences of letters in tokens). Syntactic grammars are organized sets of graphs, and are called local grammars. For instance, NooJ can be used for building grammars that are used to recognize and annotate multi-word units and semi-frozen expressions, Context-Free parsers that can compute sentence structure, Enhanced Grammars that can perform semantic analysis transformations on texts and translations, etc.

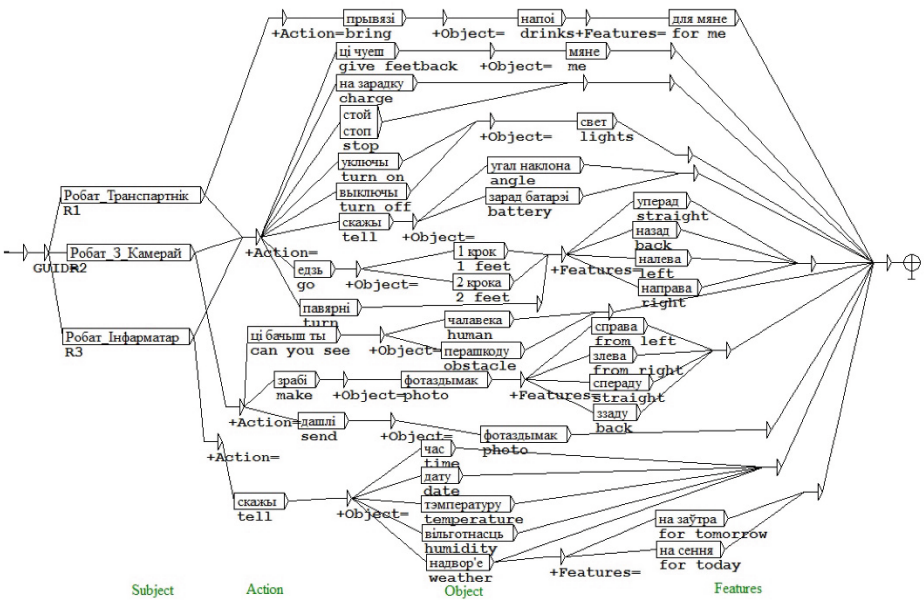

Fig. 1. FST for language modelling for three specific robots

We use NooJ Syntactic Grammar to design the transducer for combining all the concepts of our Language Model mentioned above (*'Subject'*, *'Action'*, *'Object'* and *'Features'*) and linguistic units that will refer to them.

Here is a simple example of the Language Model for the communication of a human that speaks Belarusian with a group of robots (Fig. 1). In this example each of robots can perform primitive commands, but, in group, all the robots may perform something more complex and reasonable.

Once the Language Model has been designed, we can either keep it further strictly untouched as it is predefined or we can add the possibility of developing the language. We can develop it, for example, by adding a reduced set of commands (which is natural for everyday human speech, military men, etc.) or synonyms, etc.

5 Language Generation Using NooJ Dictionary

In NooJ, *Dictionaries* usually associate words or phrases with a set of information, such as category, one or more inflectional and/or derivational paradigms, one or more syntactic or semantic properties, domain classes, etc.

NooJ dictionaries can store syntactic and semantic information (just like a lexicon-grammar) as well as multilingual information and thus can be used in machine translation systems.

We use a play-out approach to generate a dictionary of the language for robots using our FST and *NooJ Dictionary tool*. Here are some units that comprise the generated dictionary. Each unit in the dictionary below is the result of one of the paths started with 'start/in node' and ended with 'end/out node'.

An example of the generated language:

…

…

Робат _ Інфарматар скажы дату, GUID = R3 + Action = tell + Object = date

Робат _ Інфарматар скажы тэмпературу, GUID = R3 + Action = tell + Object = temperature

Робат _ Інфарматар скажы час, GUID = R3 + Action = tell + Object = time

Робат _ 3 _ Камерай дашлі фотаздымак, GUID = R2 + Action = send + Object = photo

Робат _ 3 _ Камерай зрабі фотаздымак злева, GUID = R2 + Action = make + Object = photo + Features = from right

Робат _ 3 _ Камерай зрабі фотаздымак справа, GUID = R2 + Action = make + Object = photo + Features = from left

Робат _ Транспартнік едзь 2 крока направа, GUID = R1 + Action = go + Object = 2 feet + Features = right

Робат _ Транспартнік едзь 2 крока налева, GUID = R1 + Action = go + Object = 2 feet + Features = left

Робат _ Транспартнік едзь 2 крока назад, GUID = R1 + Action = go + Object = 2 feet + Features = back

Робат _ Транспартнік едзь 2 крока уперад, GUID = R1 + Action = go + Object = 2 feet + Features = straight

Робат _ З _ Камерай скажы угал наклона, GUID = R2 + Action = tell + Object = angle
Робат _ З _ Камерай выключы свет, GUID = R2 + Action = turn off + Object = lights
Робат _ З _ Камерай ці чуеш мяне, GUID = R2 + Action = give_feetback + Object = me
Робат _ З _ Камерай на зарадку, GUID = R2 + Action = charge
Робат _ З _ Камерай уключы свет, GUID = R2 + Action = turn on + Object = lights
Робат _ З _ Камерай стой, GUID = R2 + Action = stop
…
…

6 Language Modeling Using OSTIS & NooJ Grammars

Open Semantic Technology for Intelligent Systems [4] (OSTIS) is a tool that facilitates semantic analysis by providing a unified semantic network [2] (USN) approach to knowledge representation. USNs allow knowledge engineers to represent a multitude of knowledge types within a single, unified framework using specially constructed graphs.

In this section we illustrate how to generate the same Language Model for human interaction with a mobile robot using OSTIS. We will consider the mapping of the NooJ Graph for syntactic grammar to the graph that is specific to OSTIS technology. High-level mapping model and OSTIS-based system is shown as a data flow diagram in Fig. 2.

The data flow process is as follows:

1. The user puts in a query to the system using either voice (through a speech recognition system) or keyboard. It might as well be any other input device that his system is equipped with, the only requirement being the ability to output a text in some way.
2. The query is then analyzed using the NooJ grammar, which had been converted to USN beforehand with a specialized conversion agent and stored in a shared semantic memory. Shared semantic memory is a part of a software implementation of dynamical graph models [2] provided by OSTIS technology.
3. The user input analyzer outputs up to four request components as a result, depending on the request complexity and grammar structure.
4. Request generator then builds a request instance and stores it in a shared memory as well.
5. Semantic memory notifies all the subscribed robots about a newly formed request. Notification is triggered by creating an arc from a certain request class to a certain request instance
6. Each robot decides if that request is referred to it and if so, performs the requested action with a requested object, and then leaves an execution report for other parts of the system to process.

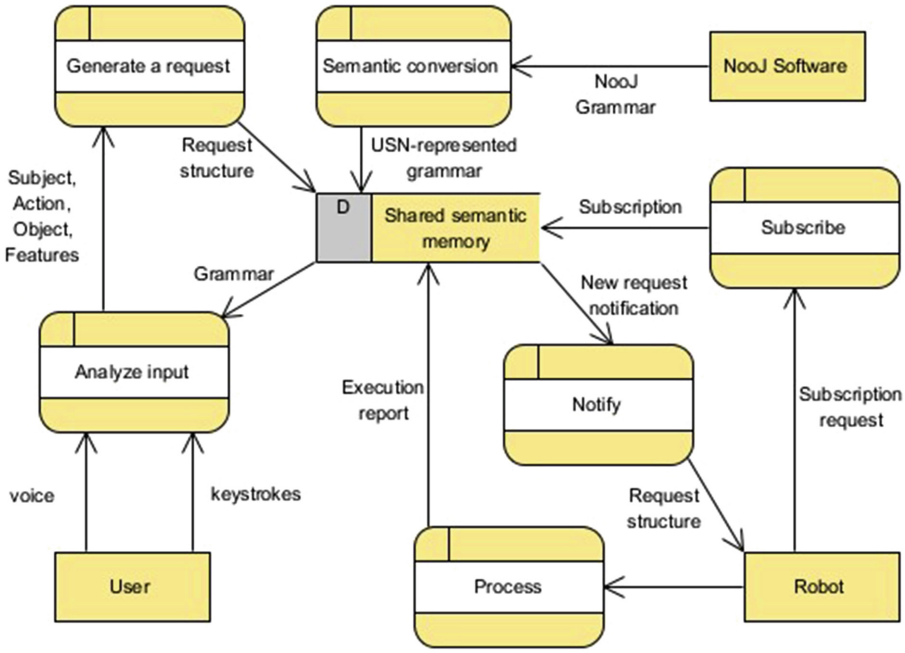

Fig. 2. OSTIS data flow diagram

The diagram above shows that the shared semantic memory plays a crucial role in the interaction, serving as an integration medium between the user and the mobile service robot.

7 OSTIS Representation of NooJ Concepts

To map NooJ graphs into OSTIS-based graphs successfully it is necessary to represent both NooJ concepts and target system's concepts and entities using USNs.

Firstly, the target system's entities and concepts need to be defined. In terms of NooJ grammars, they can be represented by the values of the +Object attribute. To define and appropriately group them, subject domains are used in OSTIS. If we consider the human-robot interaction, we have a sample subject domain called "Service Robot subject domain" which consists of entity classes, such as "human", "obstacle" and "photograph", their particular instances and relationships between them. USN representation of this subject domain's entity classes is shown in Fig. 3.

Secondly, we have to define a system of commands for our robots to understand. In OSTIS user-interface subsystem terms, they can be atomic and non-atomic. Non-atomic commands decompose into atomics and narrower non-atomics. Non-atomic commands can be thought of as the top-level menu items or items with submenus in GUI programs. Atomic commands can be thought of as the items without submenus. Even though menu metaphor is used here, this doesn't mean that actual

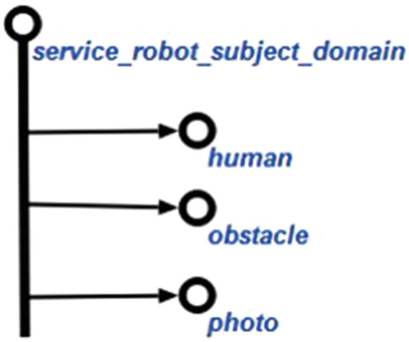

Fig. 3. Sample subject domain entities

on-screen menus must be used, since the user interface is applied here in general sense (as the means of user interaction with a technical system, not necessary graphical). Figure 4 below depicts a number of atomic commands a robot can understand (walk, turn, stop, tell, send and so forth) along with some non-atomic ones (move, report, do action, etc.). Atomic commands correspond to +Action attribute values in NooJ grammar. When a command is triggered, it initiates a request by putting a special construction in memory, which will be shown next.

When +Subject, +Object, +Action and +Features attributes are retrieved, a request structure can be built. +Action attribute becomes a request type. +Subject, +Object and +Features become its arguments. Then those are annotated accordingly. Empty node with no label denotes a specific instance of the request. Robot can subscribe to a specific event in a semantic network (node creation or deletion, arc creation or deletion from or to a specific node) via a wireless connection over a specialized network protocol. Once a structure, such as one depicted in Fig. 5, has been constructed in memory, the notification is sent to a participating robot. Then this robot determines if it can perform the request and if it does, then it leaves a report in memory for other software or hardware agents to possibly process it.

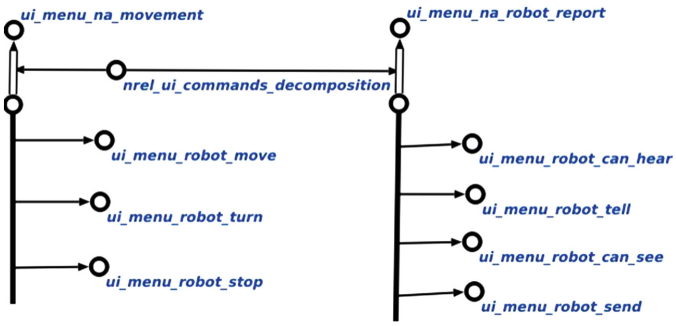

Fig. 4. USN representation of the commands

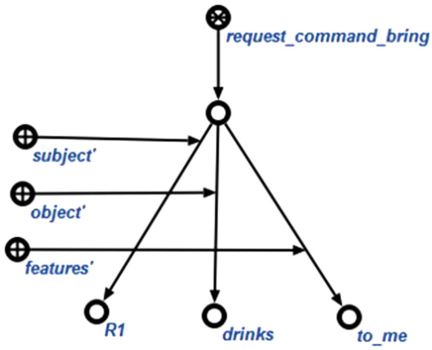

Fig. 5. USN representation of the sample request

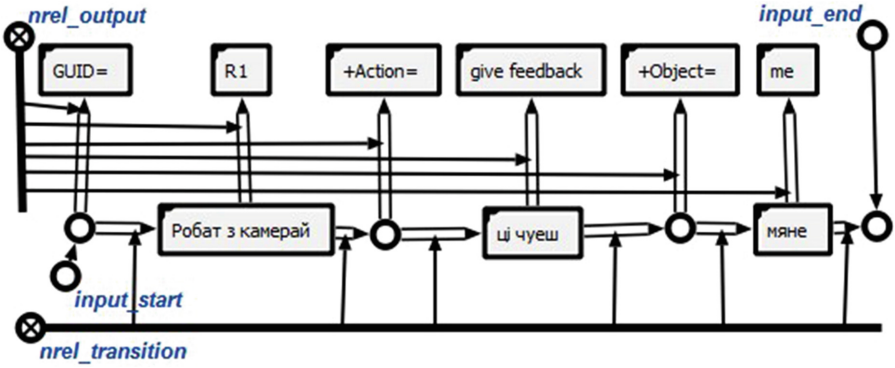

Fig. 6. USN representation of the NooJ grammar fragment

Since both technologies are graph-based, the USN representation of NooJ grammar is pretty straightforward (Fig. 6). Empty nodes with no labels denote epsilon-transitions. Wide arrows denote arbitrary connections that must be further specified. In our case we use two relations – transition and output. Narrower arrows denote an "is-a" relation (set membership or class-to-entity, to be exact) – the only relation that doesn't need to be specified. Thick dark lines denote so-called "buses" used to extend node's contact area. Text boxes denote specific input or output strings.

8 Example of OSTIS-Specific Grammar Processing

Now we can use the graph in Fig. 6 to generate a dictionary. In particular, we traverse the graph and retrieve all the inputs using a specialized language. And we only need 13 low-level (abstract graph processing machine) operators to do this. An excerpt of traversal results can be seen below in Fig. 7. Furthermore, those graphs can be programmatically transformed. And the transformation/traversal programs can transform

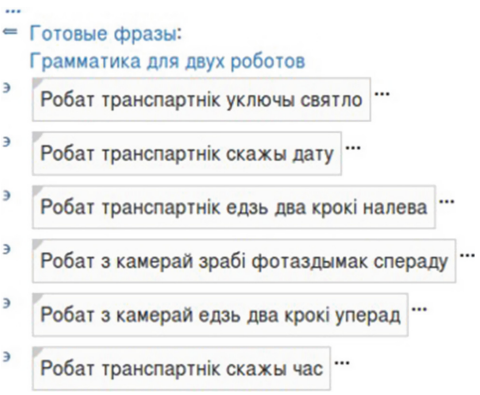

Fig. 7. Partial traversal results

itself since they are graphs as well and reside in the same memory as anything else mentioned above.

9 Conclusion

Language modeling for robots-human interaction through speech is a tricky task. On the one hand it should be simple enough to meet recognition performance requirements and on the other hand it should be close to human speech. These requirements lead us to design a language model without ambiguity. When recognizing commands, any possible uncertainty should be resolved by communication between robots and a human.

In this paper we presented the way of language modeling using SAOF concepts. We showed two different tools that could be used for this. The first one is NooJ — the development environment mainly used to construct large-coverage formalized descriptions of natural languages, and apply them to some large corpora, in real time. The second one is OSTIS – a tool that facilitates semantic analysis by providing a unified semantic network approach to knowledge representation. As we see, any of these tools can be helpful for the language modeling and dictionary generation. NooJ has simpler notation and thus is simpler to use. OSTIS claims to have universality and robustness that could be helpful when dealing with semantics though it lacks clarity in usability. The developer of the systems should decide for himself which tool to use since it is more likely to be a matter of convenience.

References

1. Silberztein, M.: Nooj Manual [Electronic resource]. Mode of access (2014). http://www.nooj4nlp.net. Accessed 31 Dec 2014

2. Golenkov, V.V.: Graphodynamical models of parallel knowledge processing. In: Golenkov, V.V., Guliakina, N.A. (eds.) Open Semantic Technologies for Intelligent Systems (OSTIS-2012). BSUIR, Minsk, pp. 23–52 (2012)
3. Yeliseyeva, O.E.: Component design of intelligent tutoring system to prepare students for centralized testing in a foreign language. In: Yeliseyeva, O.E., Rusetski, K.V. (eds.) Open Semantic Technologies for Intelligent Systems (OSTIS-2013). BSUIR, Minsk, pp. 511–516 (2013)
4. OSTIS [Electronic resource]. Mode of access (2015). http://ims.ostis.net. Accessed 30 Sep 2015