Keep it Simple: Understanding Natural Language Commands for General-Purpose Service Robots

Keep it simple: Understanding natural language commands for general-purpose service robots

Stephany Ortuno-Chanelo[1], Luis Contreras[2*], Jesús Savage[1], and Hiroyuki Okada[2]

***Abstract*— Service robots are designed to perform useful tasks for humans, which involve managing and combining a variety of skills in the form of global and local plans to solve any given task. In this work, we propose a framework to process natural language commands for general-purpose service robots where the robot should perform an arbitrary spoken command requested by a non-expert operator. Our system uses a Natural Language Processing (NLP) parser and a Conceptual Dependency (CD) builder to create CD structures that an expert system can employ to generate a global plan that the robot will execute. An extra simplification step using Large Language Models (LLM) was tested and evaluated in order to improve the accuracy of the system. Finally, our system has been tested in challenging environments in robot competitions and we have achieved promising results.**

I. Introduction

Recently, attention on service robots has been increasing, because they have been implemented in a variety of environments, from homes to restaurants or medical modules. A domestic service robot performs home tasks for humans by using a wide range of skills such as navigation in dynamic environments, object recognition, human-machine interaction, object manipulation, or a combination of these abilities.

RoboCup@Home is an international competition where service robots are tested and challenged every year to assess the state-of-the-art in the area. This competition is designed to provide standard problems for the robotics community to solve and develop technology for future domestic applications. While some tests in the competition are focused on performing specific tasks such as cleaning up some misplaced objects, organizing them in the correct place, or guiding the new guests to a free place to sit, other tasks are performed in open and unconstrained environments (such as restaurants or the competition hall); among them, one of the most challenging tests in this competition is the general-purpose service robot (GPSR) test which examines all the combinations of abilities a robot must have to perform an arbitrary spoken command requested by a non-expert operator [1] – in this context, a simple general command like "Give me an apple" needs to combine planning and navigation skills to reach the apple's location while avoiding collisions on its way, recognize the apple and grab it, then return to the initial location and find the user to deliver the object.

One of the main goals of this task lies in the complexity of the human-robot interaction process, where the objective is to achieve a natural interaction using natural speech as a connector. Despite the language being the most typical way humans have to communicate, it has high variability in both objective (pronunciation, intonation, duration, and so on) and subjective (desire enunciation, use of ambiguous or local expressions, etc) communication aspects. Our current work presents a framework aimed at generating a plan for each given command, whose core component is based on natural language processing for expert systems. This paper addresses these issues by transforming voice commands into a representation useful for expert system interpretation to build a global plan; it is divided into two general steps:

- A robot framework using LLM to simplify and process natural language commands.
- A conceptual dependency builder based on the command's syntactic information.

II. Related Work

In recent years, several works have developed alternatives to translate natural language into a set of instructions that a service robot could execute. A novel system able to map natural language into databases is presented in [2] which uses a machine learning algorithm to match speech to text. Similarly, in [3], the authors present Instruct2Act, a framework that utilizes Large Language Models to map commands into a sequence of actions.

However, as the research in the field advances the ability to interact with non-expert users is increasing. For this reason, other approaches include semantic reasoning [4], [5], which aims to interpret spoken language and has a higher level of abstraction to represent the information; semantic reasoning has the ability to infer derived facts from existing information and if data is missing from the command it could be requested by the robot.

Recent studies have tried to address this problem by using deep learning architectures [6], [7], [8] where several layers are used to learn information representation. In general, the use of CNN and RNN and their multiple combinations constitute the baseline [9] of this field, for example in [10] which proposes a sequence-to-sequence architecture with noise injection to mitigate some errors in speech recognition steps which allow us to have a more robust system.

The authors are with the [1]Biorobotics Laboratory, School of Engineering, National Autonomous University of Mexico, and [2]Advance Intelligence and Robotics Research Center, Tamagawa University.

*contreras.luis@lab.tamagawa.ac.jp.

ACKNOWLEDGMENT: This article is based on results obtained from a project, JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO) and by PAPIIT-DGAPA UNAM under Grant IG-101721.

Although deep learning is a popular approach, not only the size and quality of the dataset, but the number of computational resources and the complexity of the architectures needed to train these models are raising several concerns. Furthermore, deep models could add bias to the learning models complicating the generalization process of these systems. Some proposals to solve this last issue have risen recently; for example, in [11] the values of neural attention layers and a semantic parser are used to reduce the bias induced by training.

General-purpose command understanding for service robots has been proposed in [12] and [13], where they use large language models to create the plan from a list of specific actions. Similarly, CodeBotler + RoboEval (`https://amrl.cs.utexas.edu/codebotler/`) propose a system for the evaluation of such systems. However, these systems fully rely on LLM to create the plans (or even the code to perform such plans) but it has been demonstrated the limitations of such approaches in task-specific problems [14] and we have found it also inaccurate in natural speech commands. Instead of letting LLMs generate a plan, we have proposed a framework to parse GPSR commands using a syntactic parser to create a Conceptual Dependency (CD) structure that allows a rule-based expert system to generate a global plan that the robot could execute [15]; we extend this approach by introducing a large language model to simplify the input command by removing ambiguities and unnecessary information and rephrasing the expression to improve the semantic understanding; by doing so, we add structure to a general command that can be used later to generate plans by using complex prompts with fixed layouts as input.

III. Simplification of Natural Language Commands

Our proposed framework structure is displayed in Figure 1, where the main components of the system are displayed: a syntactic parser to extract the syntax keywords of each command, and a conceptual dependencies builder based mainly on grammatical rules. Each of these blocks will be explained in detail in the following sections.

A. Natural language processing

Given an input command from a speech-to-text system (in this paper we use Whisper [16]), we extract relevant syntactic information to create conceptual dependencies. First, we use SpaCy parser [17], a transition-based dependency parser to process natural language texts for syntactic parsing, part-of-speech tagging, and entity recognition [18], a basic example of this pipeline is shown in Figure 2.

The spaCy parser can assign morphological features through a rule-based approach. The parser can use regular expressions to identify features, such as whether a particular word is a noun or a verb. It can also rely on context clues to tell you what features to assign to a word, such as whether a particular word is the subject or the object of a verb just by inspecting the syntax of the sentence.

The following list explain some of these features.

- ADJ: adjective (red, big)
- ADP: adposition (to, in)
- ADV: adverb (there, very)
- AUX: auxiliary (is, will)
- DET: determiner (a, an, the)
- NOUN: noun (kitchen, bedroom)
- PRON: pronoun (he, she)
- VERB: verb (go, bring)
- PROPN: proper noun (Mary, Alex)

In specific, from the syntactic parser, the part-of-speech tagging was used to recognize syntax keywords, such as the verb, the subject, the object, etc (Table I).

Text tokenization	Part-of-speech tagging
Give	VERB
an	DET
apple	NOUN
to	ADP
Mary	PROPN

TABLE I: Output example for a simple command of GPSR type.

Another advantage of the SpaCy parser is that contain a visualizer that shows the dependency parse tree. These visualizations provide a deep analysis of the text, showing the relationships between different words and phrases, as well as highlighting important entities in the text. For example, a DETERMINER **a** usually is followed by a NOUN, as shown in figure 3.

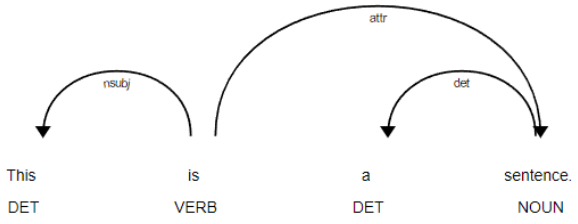

Fig. 3: Relationships between the elements in a sentence.

Taking advantage of the SpaCy library we also employed the *noun chunks* tool, which are the main noun and the words describing it.An example is shown in table II; it gives us the possibility to distinguish, for example, between a “small table” and a “tall table”, which is very common in real environments.

Text tokenization	Part-of-speech tagging
Give	VERB
a green apple	DET, ADJ, NOUN
to	ADP
red-haired Mary Jane	ADJ, PUNCT, ADJ, PROPN, PROPN

TABLE II: Output example for a command with complex nouns.

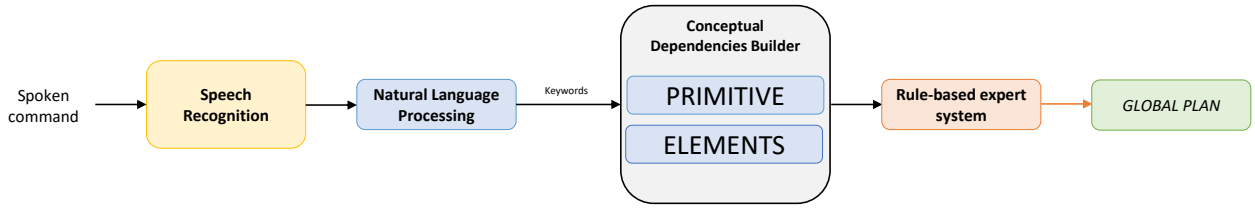

Fig. 1: Framework structure for robot plan construction from natural language commands: Speech recognition, syntactic parser, conceptual dependencies' builder, and rule-based expert system.

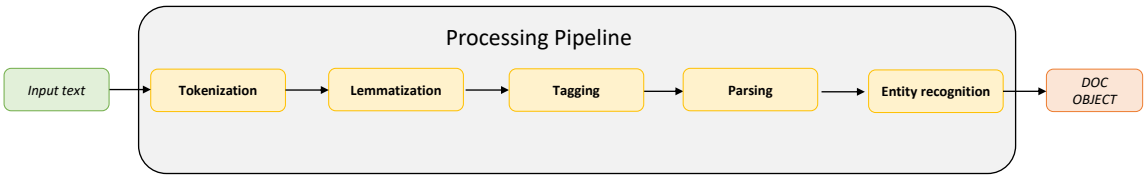

Fig. 2: Syntactic parser processing pipeline.

The input command and its corresponding syntax keywords will feed the subsequent steps and will allow us to construct the conceptual dependency structures.

B. Conceptual Dependencies Builder

Conceptual Dependency (CD) theory, proposed by [19], asserts that an action is the base of any sentence, and from the verb, we can construct a structure that describes its meaning. One of the multiple advantages, but probably the most relevant for our work, is that CDs allow the creation of a rule-based system, simplifying the use of inference rules. This fact helps to reduce the ambiguities associated to natural language. Any CD primitive contains some of the following elements: ACTOR, ACTION, OBJECT, LOCATION, STATE, and TIME. Derived from this, Schank [19] proposes a finite number of primitives that compose the key units to construct and understand more complex sentences. Some of these primitives are listed below:

- ATRANS: Transfer of possession (give, bring, deliver, etc).
- PTRANS: Transfer of the physical location (go, navigate, etc).
- ATTEND: To focus the attention on someone or something (see, meet, etc).
- GRASP: To grasp an object (get, pick, grasp, etc).

This work focuses on a CD builder that takes advantage of syntax keyword recognition to fill in the information required by each CD primitive. In order to build a complete structure that an expert system could use to infer an execution plan. Below is displayed a typical CD structure showing its constitutive elements. Where the word *nil* indicates missing information.

PTRANS((**ACTOR** nil) (**OBJECT** nil) (**FROM** nil) (**TO** nil))

The following examples are used to explain the functionality of the CD builder. Using not only part-of-speech tagging but also the relationship between them, we can extract a list of keywords and make an association with the elements in the primitive.

COMMAND: Give an apple to her.

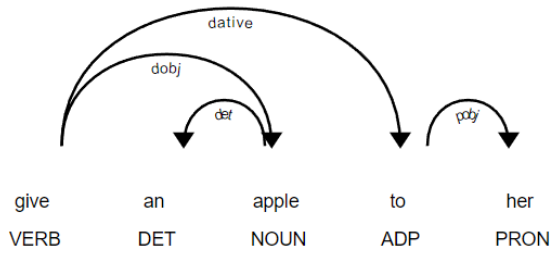

Fig. 4: Relationships between the elements of the command "give an apple to her".

Keyword	Tagging	CD element
give	VERB	ATRANS
an apple	NOUN	*obj*
her	NOUN	*person*

TABLE III: Element matching for the first command.

COMMAND: Give her an apple.

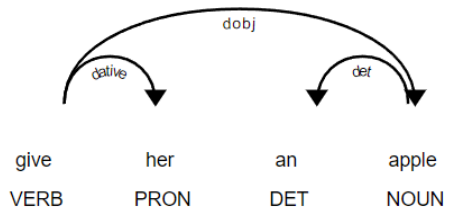

Fig. 5: Relationships between the elements of the command "give her an apple".

Keyword	Tagging	CD element
Give	VERB	ATRANS
her	NOUN	*person*
an apple	NOUN	*obj*

TABLE IV: Element matching for the second command.

Despite being described differently in natural language and with the relationships between their entities as shown in Figure 4 and Figure 5, the interpretation of the sentences represent the same command: "give someone an apple". It can be observed in Table III and Table IV that we have a different syntax keyword order but the same CD elements that fill in the primitive. As a result, from these commands, we generate the following CD structure for both of them:

ATRANS((**ACTOR** robot)(**OBJECT** an apple)(**TO** person))

Following this simple yet powerful strategy, we have implemented an efficient way to translate from a natural language command to a structure that helps to reduce these ambiguities.

IV. Experiments and Results

For the experiments, we randomly generated 50 commands using the RoboCup's GPSR command generator as in `https://github.com/kyordhel/GPSRCmdGen`. The table V includes some examples of the commands generated and the quantity of each difficulty contained in the dataset used to test the system. The easiest difficulty commands generated only required to build one simple CD structure. However, the hardest difficulty commands generated were the most challenging, which need more than one CD structure or they use a very complex language or task (ex: phrasal verbs).

Difficulty	Example	Quatity
Easy	Follow James	21
Medium	Please go to the exit, meet Jennifer and guide her	11
Hard	Could you deliver cutlery to the person pointing to the left in the bedroom	18

TABLE V: Classification of the commands generated by difficulty.

In this work, we assume that the commands have been recognized correctly by the speech recognition system. Mainly because the system described in this work is not interested in the recognition or planning step but in the syntactic reasoning of commands that can be used to make complex plans or any other type of reasoning. For instance, we want to create a system that can process commands such as "go to the bedroom" and "bring the groceries". We want to be able to reason about the logic behind the commands, such as what items are involved in the command, where to find them, and how to carry out the task.

Consequently, we will focus on the command syntactic analysis to generate conceptual dependencies as follows.

A. Conceptual Dependencies (CD) builder

We generated 50 commands and evaluated the outputs from the syntactic parser and CD Builder. Figure 6 shows the accuracy of the system by each level of difficulty. Where the system achieved 95% for the easiest level, 54% for the medium level and 72% for the hardest level. Leading to a global accuracy of 82% in tagging the elements and making the CD structures.

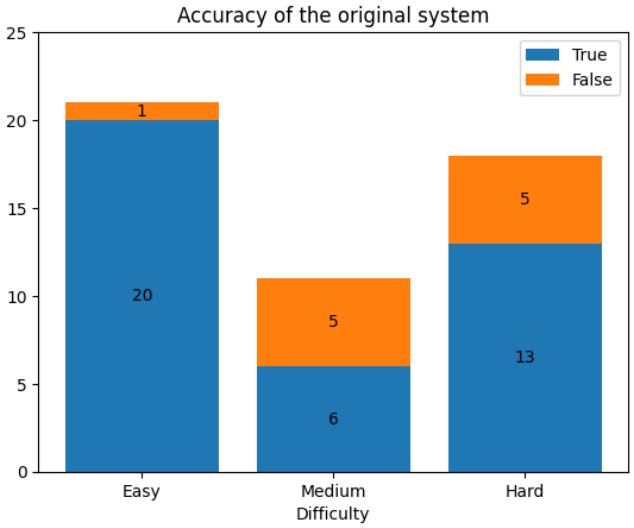

Fig. 6: Bar graph showing the accuracy of the system by level of difficulty: In blue the number of successful recognition, in orange the amount of failed attempts.

The following examples illustrate the complexity of the commands generated and the outputs obtained by the system.

COMMAND (easy): Deliver a red apple to me

OUTPUT:

ATRANS((ACTOR robot) (OBJECT red apple) (FROM red apple place) (TO me))

This command implies a change in an object's property. The output indicates that the robot needs to take the red apple from its original location or ownership to another person. As we can see, here we have a complex noun: "red apple" which the system could recognize successfully.

COMMAND (hard): Could you please navigate to the end table, meet Robert and lead him.

OUTPUT:

PTRANS ((ACTOR robot) (OBJECT robot) (FROM robot place) (TO end table))

ATTEND ((ACTOR robot) (OBJECT Robert)

PTRANS ((ACTOR robot) (OBJECT Robert) (FROM Robert place)(TO nil))

Here we have a medium difficulty command which requires more than one CD structure. The output indicates that the robot will move to a new location, find a person and then lead that person somewhere (not mentioned).

Some of these errors are shown below.

COMMAND (medium): Meet Alex and take her.

OUTPUT:

GRAB ((ACTOR Robot) (OBJECT Alex))

As we can observe in this example, there are some verbs in specific contexts that the system is unable to recognize. Another command that presented an interesting challenge was the following.

COMMAND (hard): Could you deliver cutlery to the person pointing to the left in the bedroom?

OUTPUT:

ATRANS ((ACTOR robot) (OBJECT cutlery) (FROM cutlery place) (TO person))

From this example, we can see that our system has some limitations when the noun description is very complex (the person pointing, the person waving, etc). Some other areas of opportunity are also present, as the GPSR command generator uses some unusual verbs that could present problems for our system.

B. Large Language Models (LLM) for command simplification

Taking into account the results of the previous section, it is evident that with more sophisticated nouns and commands we need an extra step to solve these kinds of complications or ambiguities. Therefore, we propose using LLM to rephrase the input command to a more structured sentence useful for conceptual dependencies' construction.

In our experiments, we use ChatGPT [20], a natural language processing (NLP) system developed by OpenAI, because it is a powerful NLP system able to understand the context of human conversation and generate appropriate responses.

Using LLMs provides the opportunity to robust the human-robot interaction [21]. Large language models can be used to rephrase the original command using a specific range of word options and avoiding ambiguities and redundant language in order to obtain a simplified version of the command which our CD builder could manage in a more efficient way.

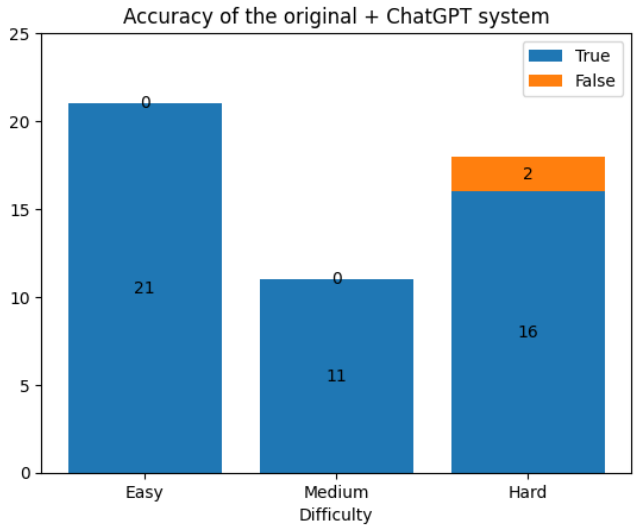

Fig. 7: Bar graph showing the accuracy of the system + ChatGPT by level of difficulty: In blue the number of successful recognition, in orange the amount of failed attempts.

Figure 7 shows the accuracy of the system by each level of difficulty. Where the system achieved 100% for the easiest level, 100% for the medium level and 88% for the hardest level. Leading to a global accuracy of 96%.

Some of the results obtained are explained below.

COMMAND SIMPLIFICATION:
Could you deliver drinks to the person raising their left arm? $\rightarrow$ Deliver drinks to the person raising their left arm.

OUTPUT:
ATRANS ((ACTOR robot) (OBJECT drinks) (FROM drinks place) (TO person raising their left arm))

In this example, it is possible to see that an LLM allows simplifying the command from a high level of difficulty to a more appropriate and simplified version that the CD Builder could recognize and manage without inconvenience.

A second example that uses unusual verbs to describe the action is shown below.

COMMAND SIMPLIFICATION:
Meet Alex and take her. → Meet Alex and guide her.

OUTPUT:
ATTEND ((ACTOR robot) (OBJECT Alex)
PTRANS ((ACTOR robot) (OBJECT Alex) (FROM Alex place)(TO nil))

This command implies finding and leading a person, but it uses the verb *take* as a synonym of *guide*. In this example, it is possible to observe that the LLM makes a new command using the verbs and expressions included in the CD builder to rephrase the command without changing its meaning. Adding this extra step and rephrasing the commands using LLM the system obtains a 96% of accuracy in the original dataset, as shown in Figure 7.

However, there are some commands that could not be understood, like the following:

- Take out the garbage. → Guide the garbage out.
- Describe the objects on the table. → Find the objects on the table and describe them.

Finally, our framework, including the CD builder, was tested on competition environments showing promising results and contributing to winning 1st place in the RoboCup Mexican Open 2023, 2nd place in the RoboCup Japan Open 2023 and 6th place at the international RoboCup 2023 in the at Home league.

V. Conclusions

In this work, we have shown the use of conceptual dependencies to understand natural language commands and make plans, and we have improved the performance by simplifying the input command using large language models (LLM) rather than allowing the LLM to generate the plan.

This approach allows us to deal with complex nouns, where we can include a description of the object like *red apple* or *end table*, which helps to distinguish between the options present in real and more challenging scenarios.

From our results, we observe that using a syntactic parser and a CD builder gets an 82% accuracy at recognizing a range of general-purpose commands from the RoboCup's GPSR command generator while using LLM to simplify the input command helps to increase the performance of our system to a 96% accuracy.

Although LLMs are powerful systems, they have certain limitations because it is trained on large datasets that may produce responses that contain biased especially when there are several ways to interpret a command.

These errors in interpretation include "phrasal verbs" which are more common in informal contexts; phrasal verbs contain a particle that often changes the meaning of the verb.

Other errors are related to the limitations of our system where, for example, we have not considered a conceptual dependency that fits the *describe* action in the "describe an object" command.

References

[1] J. Hart, M. Matamoros, A. Moriarty, H. Okada, M. Leonetti, A. Mitrevski, K. Pasternak, and F. Pimentel, "Robocup@home 2022: Rules and regulations." `https://athome.robocup.org/rules/2022_rulebook.pdf`, 2022.

[2] A. Giordani, "Mapping natural language into sql in a nlidb," pp. 367–371, 06 2008.

[3] S. Huang, Z. Jiang, H. Dong, Y. Qiao, P. Gao, and H. Li, "Instruct2act: Mapping multi-modality instructions to robotic actions with large language model," 2023.

[4] X. Chen, J. Ji, J. Jiang, G. Jin, F. Wang, and J. Xie, "Developing high-level cognitive functions for service robots," vol. 2, pp. 989–996, 01 2010.

[5] Y. Bisk, D. Yuret, and D. Marcu, "Natural language communication with robots," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (San Diego, California), pp. 751–761, Association for Computational Linguistics, June 2016.

[6] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," 2018.

[7] J. Chai and A. Li, "Deep learning in natural language processing: A state-of-the-art survey," in *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, pp. 1–6, 2019.

[8] H. Li, "Deep learning for natural language processing: advantages and challenges," *National Science Review*, vol. 5, pp. 24–26, 09 2017.

[9] D. Fried, J. Andreas, and D. Klein, "Unified pragmatic models for generating and following instructions," 2018.

[10] Y. Tada, Y. Hagiwara, H. Tanaka, and T. Taniguchi, "Robust understanding of robot-directed speech commands using sequence to sequence with noise injection," *Frontiers in Robotics and AI*, vol. 6, 2020.

[11] M. Mensio, E. Bastianelli, I. Tiddi, and G. Rizzo, "Mitigating bias in deep nets with knowledge bases: The case of natural language understanding for robots," *CEUR Workshop Proceedings*, vol. 2600, May 2020. AAAI 2020 Spring Symposium on Combining Machine Learning and Knowledge Engineering in Practice, AAAI-MAKE 2020 ; Conference date: 23-03-2020 Through 25-03-2020.

[12] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "Tidybot: Personalized robot assistance with large language models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[13] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (London, UK), 2023.

[14] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati, "Large language models still can't plan (a benchmark for llms on planning and reasoning about change)," 2023.

[15] A. Presacco, J. Savage, J. Cruz, and S. Ortuño, "A framework to process natural speech in service robots using a combination of a speech recognition system, universal dependencies extracted by means of the stanford parser and an expert system," in *Interactive Collaborative Robotics* (A. Ronzhin, G. Rigoll, and R. Meshcheryakov, eds.), (Cham), pp. 172–181, Springer International Publishing, 2021.

[16] "Whisper." `https://openai.com/research/whisper`. Accessed: 2023-05-30.

[17] "Spacy parser." https://spacy.io/api/dependencyparser. Accessed: 2023-04-05.

[18] V. Okhapkin, E. Okhapkina, A. Iskhakova, and A. Iskhakov, *Constructing of Semantically Dependent Patterns Based on SpaCy and StanfordNLP Libraries*, pp. 500–512. 03 2021.

[19] R. Schank, "Conceptual dependency: A theory of natural language understanding," *Cognitive Psychology*, vol. 3, pp. 552–631, 1972.

[20] "Chatgpt openai." https://openai.com/chatgpt. Accessed: 2023-06-18.

[21] Y. Ye, H. You, and J. Du, "Improved trust in human-robot collaboration with chatgpt," *IEEE Access*, vol. 11, pp. 55748–55754, 2023.