On Evaluating LLM Integration into Robotic Architectures

VASANTH SARATHY, MARLOW FAWN, MATTHEW MCWILLIAMS, and MATTHIAS SCHEUTZ, Tufts University, USA
BRADLEY OOSTERVELD, Thinking Robots, USA

LLMs are being increasingly integrated into embodied robotic systems. A useful capability that the LLMs bring to robots is translating noisy spoken human natural language instructions into executable robot actions. However, these integrations are somewhat ad-hoc and understudied as they tend to not consider the gamut of syntactic, semantic, as well as, pragmatic aspects of embodied human communication. What is missing is a characterization of the different paradigms for integrating LLMs into robotic architectures as well as a set of evaluation metrics that capture whether an LLM-equipped robot can correctly understand these different aspects of human instruction. In this paper, we present a suite of evaluation metrics together with data augmentation techniques for evaluating these architectures, using concepts from the cognitive science and human communication literature. To illustrate an application of these metrics and augmentation techniques, we conduct experiments to to compare two integration methods: LLMs as pre-processing components that map human instructions into more constrained versions to be processed by the architecture's natural language understanding (NLU) subsystem, or LLMs as a wholesale replacement for the NLU's parser. We provide experimental evaluations and a robotic implementation to show the inherent tradeoffs between the methods. Our results suggest that while they offer increased explainability, traditional parsing tools coupled with LLMs do not perform as well as an LLM that replaces a parser entirely. The proposed evaluation metrics together with the characterization of different LLM integration approaches offer the promise of systematically evaluating LLMs as natural language interfaces to robotic systems as well as tackle the important tradeoff between explainability/verifiability/interpretability and robustness to noisy input and broad language understanding in an open-world embodied setting.

CCS Concepts: • **Computing methodologies** → *Discourse, dialogue and pragmatics*; **Cognitive robotics**.

Additional Key Words and Phrases: large language models, cognitive architecture, semantic parsing, robotic architectures, evaluation metrics

1 INTRODUCTION AND MOTIVATION

For robots to become useful to non-experts they need to be instructible in natural language as natural language is the most natural form of human communication: we simply want to be able to talk to robots and tell them what to do and how to do it. However, human language is complex and often accompanied by a range of unique syntactic, semantic and pragmatic challenges. Take for example the utterance:

"*Can you... um please get mug, no, I mean class... on that table.*"

This utterance contains a number of features that have historically been deep research challenges, in and of themselves, including *indirectness* ("can you..." question structure for a request), *informality*, improper grammar from potentially *non-native speakers* ("...get mug..."), *disfluencies* ("um"), *corrective language* ("..no I mean...")

Authors' addresses: Vasanth Sarathy, vasanth.sarathy@tufts.edu; Marlow Fawn, marlow.fawn@tufts.edu; Matthew McWilliams, matthew.mcwilliams@tufts.edu; Matthias Scheutz, matthias.scheutz@tufts.edu, Tufts University, 177 College Ave, Medford, Massachusetts, USA, 02155; Bradley Oosterveld, brad@thinkingrobots.ai, Thinking Robots, 12 Channel Street, #202, Boston, Massachusetts, USA, 02210.

ACM 2157-6912/2025/8-ART
https://doi.org/10.1145/3754340

and a range of issues when converting speech to text (e.g., words can drop off or are misheard – e.g., "class" should have been "glass"). With robots, these challenges are further exacerbated by situated and embodiment concerns like the limits of a robot's own capabilities (e.g., the robot needs a mug detector and the ability to pick up objects and potentially even navigate to the table), environmental constraints such as whether mugs and glasses on tables actually exist in the current environment, and referential considerations that require interpreting diectic references ("that table" versus say "a table"). Thus, a central problem of a robot's language pipeline is to understand human instructions and intent, in the presence of lexical ambiguity, variable syntax, and pragmatics in embodied robotic contexts.

To solve this problem, traditionally, robotic architectures have employed symbolic semantic parsers such as combinatorial categorial grammars (CCG) to parse utterances and extract their true intended meaning in terms of functions in the robot's action and perception repertoire [5, 21]. However, they can be brittle and require an extensive grammar to flexibly parse different utterance forms. More recently, with the advent of Large Language Models (LLMs), robotic systems have been better able to handle natural human speech [2, 24, 27]. This line of work has looked to employ LLMs wholesale in robotic architectures to handle not only natural language understanding but also planning and reasoning. Despite impressive results, it is unclear whether these systems can handle the types of pragmatic and semantic challenges noted above. There are no suitable evaluation metrics or datasets to better characterize the performance of the pragmatics understanding capabilities of LLMs in these robotic settings. Moreover, their end-to-end nature reduces the degree of interpretability and verifiability of these systems. Thus, the crucial tradeoff in natural language understanding for robotic systems is between the ability to handle utterance diversity on the one hand, and verifiability on the other. Recent work in code generation and executable language grounding [10, 16, 26] has explored potentially combining LLMs with formal systems to extract programs and formal expressions from natural language. However, they do not tackle the issues beyond basic reference resolution, which, albeit is a rich and deep problem, in and of itself.

In this paper, we explore whether robotic systems equipped with both LLMs (capable of providing robustness to variable and noisy input) and a more traditional symbolic cognitive architecture (capable of providing verifiability and transparency) are able to handle these stated challenges. We develop novel evaluation techniques (metrics and data augmentations) that target the performance of LLMs in robotic systems. We use these evaluation techniques to identify the pros and cons associated with two different integration approaches: LLM as a parser vs LLM as a translator. In the parsing approach (a deeper integration), the robot turns the utterance into a functional form that is readily usable in the architecture. In the translation approach (a shallower integration), the robot turns the incoming utterance into a "simpler" form that is then passed to a traditional CCG-type parser, which, in turn, extracts the semantics and pragmatics. Underlying these approaches is a semantic parse structure ("information structure") that is founded on decades of theory on human communication, pragmatics, and cognitive science. Our evaluation metrics are based on this foundational notion of "information structure".

Our specific contributions are: (1) an automated approach to augmenting training data to account for the natural diversity encountered in human task instructions, (2) a set of metrics for evaluating natural language understanding based on speech act theory and pragmatics of information structure, (3) an implementation of the metrics and augmentation techniques to compare two integration approaches of LLMs into a cognitive robotic architecture, including their ability to handle novelty in the open-world, and (4) a general paradigm for analyzing approaches to integrating LLMs into robotic architectures.

2 PARADIGMS FOR INTEGRATING LLM(S) INTO ROBOTIC ARCHITECTURES

There are several ways in which LLMs could be integrated into robotic architectures (from least to most intrusive):

(1) **LLM-as-a-Translator:** The LLM is added as a "pre-processing component" that translates free natural language into modified (and restricted) natural language from which the system can use its existing

symbolic parsing capabilities to understand the input utterances. This has the advantage that no changes to the architecture is required, only a mapping from broad to narrow natural language is needed (e.g., simple English), and that any potential existing formal guarantees about its operation are retained. It has the disadvantage that the built-in language processing system will still limit the kinds of instructions the the system as a whole can handle.

(2) **LLMs-as-a-Component:** The LLM replaces one or more natural language processing components such as the parser, reference/anaphora resolution, etc. but also potentially other components like the planner, action sequencer, reasoner, etc. in the architecture, but the overall architectural layout, component interfaces and interactions are kept in place. This has the the advantage that the overall architecture with LLM component replacements should work as before but with improved performance in some settings compared to the original version. It poses the challenge to ensure the desired component function is still achieved and not altered (e.g., that the LLM is able to generate successful plans in those cases where the previous planner would find a solution), which could impact formal guarantees on instruction interpretation.

(3) **LLM-as-an-NL system:** The LLM replaces the whole natural language subsystem of the architecture, while keeping all other components (e.g., perceptual, planning, reasoning, and action sequencing subsystems) in place. This has the advantage of utilizing the complete understanding ability of LLMs, but poses the challenge to generate appropriate outputs that the remaining architecture can handle (e.g., goal predicates, actions, plans, etc.) and it will, in general, void any formal guarantees about deterministic component operation due to the probabilistic nature of the LLM (e.g., guarantees that the given the same context, the same plan will be generated).

(4) **LLM-e2e:** The LLM replaces all cognitive parts of the architecture, keeping only the perceptual and actuation components in place. This has the advantage of utilizing the complete natural language understanding and common sense reasoning ability of the LLM, but presents the challenge of generating input from the sensory side that the LLM can understand and output that the actuation components of the robot can execute, possibly overwhelming the LLM with large data streams, and again, removing any performance guarantees that might have existed in the architecture.

In this paper, we focus on evaluating options 1 and 2, i.e., we compare the performance of front-end LLM translation (*LLM-as-a-translator*) with LLM replacements of part of the architectural natural language processing pipeline (*LLM-as-a-Parser*). Further comparisons for options 3 and 4 will be explored in future work.

2.1 Information Structure: The *Interlingua* of Robotic Architectures

Understanding natural language requires the listener to unify the speaker's words across three separate contexts: (1) situated context (e.g., what objects are being referred to in the listener's physical environment, past dialog, etc.), (2) the speaker's goals (e.g., at a more abstract level, what does the speaker want the listener to do: perform an action, answer a question, or take a mental note), and (3) listener's capabilities (e.g., can the listener actually do what it thinks the speaker is asking it to do). The speaker facilitates this unification process by organizing their message in an informational structure [3].

An *informational structure* is an organization that reflects the contents and purpose of the information contained in the utterance. Despite differing definitions of information structure, the cognitive science literature generally agrees that an utterance's information structure has three parts: (1) previously known information (givens or supplemental propositional content (SPC) that refers to expressions, descriptors of entities and cues to attentional/cognitive status of entities in the minds of the listener), (2) new information (new or the core propositional content (CPC) of the utterance, including action imperatives that the speaker would like the listener to take or conceptual notions that the speaker would like the listener to come to believe), and (3) what the speaker

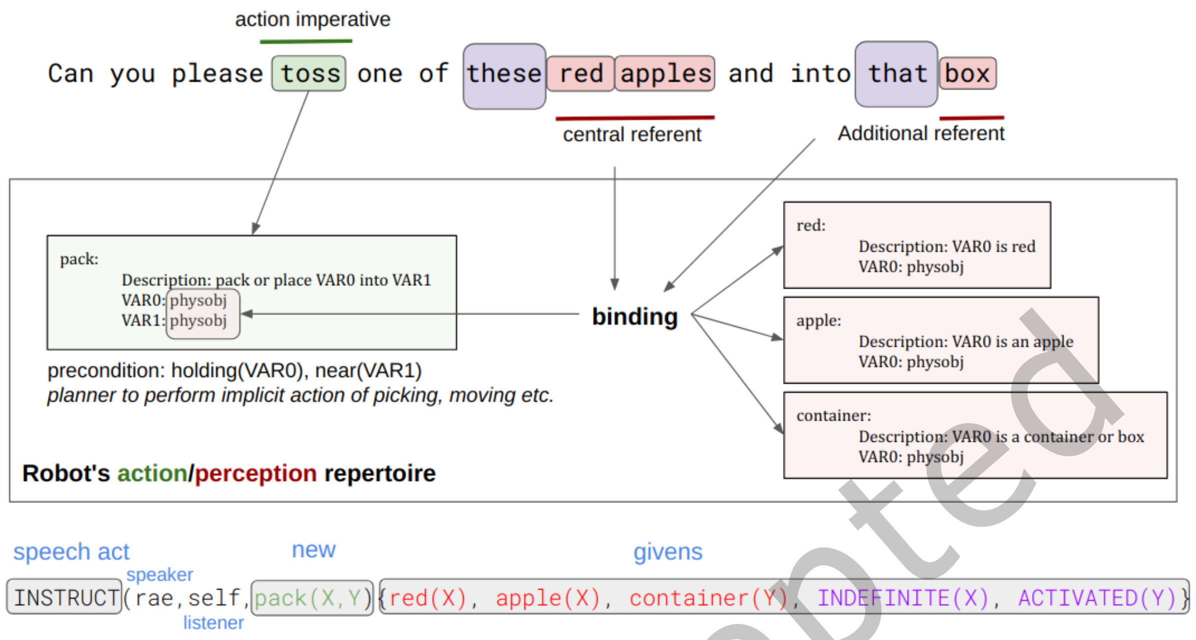
action imperative
Can you please toss one of these red apples and into that box
central referent
Additional referent
pack:
Description: pack or place VAR0 into VAR1
VAR0: physobj
VAR1: physobj
binding
red:
Description: VAR0 is red
VAR0: physobj
apple:
Description: VAR0 is an apple
VAR0: physobj
container:
Description: VAR0 is a container or box
VAR0: physobj
precondition: holding(VAR0), near(VAR1)
planner to perform implicit action of picking, moving etc.
Robot's action/perception repertoire
speech act
new
givens
speaker
listener
INSTRUCT(rae, self, pack(X,Y){red(X), apple(X), container(Y), INDEFINITE(X), ACTIVATED(Y)}

Fig. 1. Understanding a natural language imperative requires extracting (1) the speech act or underlying intent (e.g., "INSTRUCT" for a request to perform an ontic action), (2) the new information or core propositional content (CPC), which could be an action or a new concept (e.g., "pack"), and (3) the given knowledge used for reference resolution or supplemental propositional content (SPC, shown in red) (e.g., "red", "apple", "container"), as well as cognitive status indicators (shown in purple) (e.g., "that" is a deictic referring expression and maps to the cognitive status of "ACTIVATED" in the Givenness Hierarchy [11]. Crucially, our approach allows for extracting these aspects of an utterance into a lifted expression, and grounding them in the robot's own action and perceptual repertoire. "Understanding" involves the robot being able to make sense of the utterance in an actionable and groundable way.

intends for the listener to do with this information (intent): performing actions, remembering facts, answering questions, etc.

Thus, understanding natural language is, in a sense, extracting an actionable information packet that is intended by the speaker to be acted on in a particular way (see Figure 1). It is actionable in the sense that from the listener's perspective, it references actions that the listener can perform or concepts the listener can recognize and represent. The parse captures what the speaker intends for the listener to do with the information. The core challenges involve mapping imperatives to executable actions[1], and resolving referring expressions and understanding a variety of linguistic forms. The core question we explore is whether and how LLMs can assist in extracting an information structure from messy natural language.

2.2 Illustrative Hypothesis

To better elucidate our evaluation approach, we focus on a specific illustrative hypothesis. We are interested in the tradeoff between performance quality and explainability/interpretability. Given the successes of LLMs in machine translation applications, we expect that the translator approach is likely to perform just as well, if not

[1]Sometimes the actions can be epistemic in nature and might require the robot to "come to believe" a certain fact, which is still a function that must be executed by the robot.

```
put the blue block right of the yellow block
could you please place the blue block to the right of the yellow block?
stick the blue block to the right of the yellow one
uh, put the, um, the blue block, right of the, uh, yellow block
blue block, you put right side of yellow block, yes?
place the azure brick to the right of the golden brick
put the blew block write of the yellow block
put the red block, actually i mean the blue block, right of the yellow block
```

```
() = putrightof[""](Symbol ?obj:physobj, Symbol ?referent:physobj) {
    Symbol ?location:location;
    conditions : {
        pre infer : holding(?actor, ?obj);
    }
    effects : {
        success infer : free(?actor);
        success infer : not(holding(?actor,?obj));
        success infer : rightof(?obj, ?referent);
        success infer : leftof(?referent, ?obj);
    }
    op:log(info, "Putting right of");
    act:startVisualSearch(?referent);
    act:getTokenIds(?referent);
    act:moveObjectAbove(?obj, ?referent, "arm");
    act:moveObject("obj", right);
    act:moveObject("obj", down);
    act:openGripper("arm");
    act:moveObject("obj", up);
    act:goToStartPose(true);
}
```

```
{
"intent": "INSTRUCT",
"central_proposition":
"putrightof(self:agent,VAR0,VAR1)",
"supplemental_semantics": [
       "blueblock(VAR0)",
       "yellowblock(VAR1)",
       "DEFINITE(VAR0)"
       ]
}
```

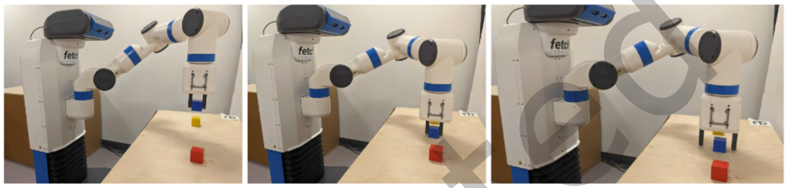

Fig. 2. Eight different variations of an imperative (top left) that is mapped into an speech-theoretic information structure (bottom left) that contains a lifted representation of the utterance capturing speaker intent, new information and given references together with a cognitive status marker. The robot is equipped with actions including the "correct" action that can be used for planning (top right). For example, here, the robot must first perform the pickup action before it can perform a putrightof action.

better than the parser approach in terms of parsing ability and quality. Because the translator approach relies on a symbolic parser, with a known grammar, it is more explainable and interpretable than the parser approach which does not rely on such a symbolic parser.[2] Thus, we anticipate that if the translation approach performs better, then it would be preferred for more trustworthy robots.

In addition to different LLM-integration approaches, we are also interested in exploring the degree to which an approach can handle novel information. Robotic systems capable of instruction-based learning of novel procedures must be able to quickly and effectively repurpose these learned actions in subsequent interactions. Our hypothesis is that LLMs provided with a set of available actions and properties as context will successfully be able to choose these actions, even if they are newly learned and unseen in the LLMs' training data. Thus, in addition to different integration modes (parser vs translator), we also explore LLMs with and without context. We expect the LLMs with context to be able to handle novel actions better.

Finally, given the recent trend towards smaller models and their success in targeted domains, we considered different model sizes. Our hypothesis is that while a larger model is likely to perform better, smaller models with additional contextual information may be able to match them.

3 EVALUATION APPROACH AND EXPERIMENTS

Broadly, our experiment consists of generating a novel dataset, finetuning various models (parsers and translators, with and without context), and evaluating the models on two hold-out test sets (familiar vs novel actions/properties).

[2]We accept that explainability, interpretability, transparency, and verifiability have varying definitions and are subject to extensive debate. Here, we stipulate for the sake of brevity that grammars and associated symbolic parsing algorithms are human-readable and explicit, which makes them more explainable or interpretable.

3.1 Data Generation

We first generated a set of 300 simple imperatives and statements together with parsed information structure, where the actions and properties are groundable in a Fetch robot [29] equipped with a cognitive robotic architecture. From a perception standpoint, the robot can perceive a set of predefined properties (e.g., object classes such as "mugs(X)" and relational properties "on(X,Y)" etc.). From an action standpoint, the robot is capable of executing a set of functions that result in joint movements (e.g., putleftof(X,Y) puts object X left of object Y). These perceptual properties and action capabilities are at the core of the robot's repertoire. We then augmented this set of utterances with style variations to generate our training and test sets.

3.1.1 Generating Base Utterances and Ground Truth Semantics. To generate each base utterance, we first define a list of all objects, prepositions, actions, and properties we want to use in the dataset. We implemented these definitions within the robotic architecture. For this work, we selected the DIARC architecture (similar to a ROS-node-based approach) because it is component-based and easily configurable, but the approach could be used with any other robotic architecture equipped with a parser [14, 18, 21]. The objects, prepositions, and properties are inserted into a parser grammar in the NL pipeline. Action information was implemented within the architecture as functions that call lower-level robot functions for manipulating joints via a kinematics planner. We use an utterance template of the form: "[ACTION] [ARTICLE] [OBJECT] [PREPOSITION] [ARTICLE] [OBJECT]" to generate imperatives and "[OBJECT] is [PROPERTY]" to generate statements. For example: "put the mug left of the banana", or "the banana is Leah's". We then filtered the utterances manually to remove those that did not make sense. We then run each of these generated utterances through the robot's current parser-based NL pipeline (which uses a CCG/lambda calculus-based parser [4, 7, 9, 25]) to produce semantics that contain the core elements of an informational structure, namely intent, new, and givens. Our initial base dataset contains 300 utterances (100 for a hold-out test set) together with the ground truth semantics and names of actions and properties available to a robot. Template-based generation has long been used in computational linguistics to ensure control and interpretability in evaluation. Our approach aims to balance template simplicity with representational breadth.

3.1.2 Data Augmentation. We then augment this dataset using an LLM (GPT-4) to generate a variety of style variations for each utterance. Specifically, we implement prompts for 8 different style variations. That is we prompt the LLM to generate a variation of the base utterance conforming to the particular style while retaining the base utterance's meaning. This then allows us to use the existing ground truth for the new utterances. Specifically, for each base utterance, we used GPT-4 to generate eight stylistic variants: (1) polite/indirect, (2) informal/casual, (3) non-native phrasing, (4) lexical substitution, (5) ASR errors, (6) disfluencies, (7) self-correction, and (8) unmodified. Prompt templates were crafted to preserve the original intent while modifying surface structure. For example, a prompt for indirectness might say: "Rewrite the utterance in below . . . to make it more indirect and polite." The full prompt templates will be made available upon publication.

The style-based augmentations include:

(1) **Directness:** makes the utterance more indirect and polite.
(2) **Formality:** makes the utterance sound more casual like a post on X (formerly, Twitter) or Reddit.
(3) **Familiarity:** rewords the utterance to appear as if the speaker is not a native English speaker.
(4) **Word Choice:** replaces the verbs and nouns with synonyms, hypernyms and hyponyms.
(5) **ASR Errors:** replicates errors that typically arise from ASR systems - word deletions, substitutions, additions. For this we used speech-generation and ASR tools to convert text to audio and then back to text.
(6) **Disfluency:** introduces "ums" and "uhs" and other disfluencies into the utterance.
(7) **Correction:** introduces inaccuracies that the speaker themselves correct mid-sentence.
(8) **No Style:** no modifications were applied.

Our final training dataset contained over 22,000 items derived by generating 10 variations of these 8 styles for each utterance, and then removing any duplicates as they arose. We then reformatted the dataset for finetuning and introduced an instruction for generating JSON output together with an example in the prompt.

3.1.3 Known and Novel Test Data. We generated a "known" hold-out test dataset in the same fashion as above. The test data consisted of 2000 utterance variations unseen in the training data, along with ground truth semantics. In a similar way, we also generated a "novel" hold-out test dataset (2000 utterances) which, unlike the familiar test set, contains actions and properties that were not anywhere in the training data utterances or in the training context. To be clear, novel utterances were constructed using action and property symbols explicitly excluded from the training data. These symbols were newly added to the robot's repertoire and used to populate utterance templates. This approach allowed us to test how well models generalize to unseen lexical-semantic combinations in an out-of-distribution setting. The novel test set allowed us to test the value of providing actions and properties as inputs together with the utterance. The idea is that if the parser is provided with novel actions and properties, it should use them. We understand that there may be other approaches to ensure that the parser uses external knowledge (e.g., Retrieval Augmented Generation, Constraint-Guided Decoding), but our goal was limited to evaluating the finetuned LLM itself to handle novel actions.

3.2 Evaluation Metrics

We are looking to generate parses that are groundable in a robotic architecture and can be executed by an embodied robot platform. We find that current popular evaluation metrics from machine translation (ROUGE, BLUE, etc.) are not suitable here. Specifically, we do not need exact string matches or even semantic similarity for the entire utterance. We need perfect matches for certain keywords (e.g., action or detector names), but we can tolerate entirely different variable names. Along the lines of work in code generation and text-to-sql and other techniques in executable language grounding [6, 10, 32], we develop a set of custom evaluation metrics for information structure parsing. Below are set of metrics capturing the different information-structure-theoretic components of parse (SPC, CPC etc.) and their relationships (e.g., using graph-matching) to ensure that the parse produced is executable, groundable, variable matching makes sense, and the degree to which different descriptors and referents are captured in the parse aligns with meaning of the utterance.

(1) **valid-json:** A binary metric indicating if the output was syntactically correct JSON, which we implement using a string literal evaluation functionality in python. This metric is practically important to ensure that LLM output can be directly used by the remainder of the architecture.
(2) **intent-correct:** A binary metric for evaluating if the intent was correctly identified as an INSTRUCT or a STATEMENT, which we implement using string matching. Although we select two intents, we envision that this metric can readily be adapted to a range of dialog acts like QUESTION, WARNING, GREETING etc. This metric is focused on the dialog level or speech-act-level analysis of the utterance, and is particularly relevant when the surface form of an utterance is different from the underlying intent. Such a discrepancy is a common phenomena especially when politeness norms are involved – interlocutors frequently use a question form to instruct an action (e.g., "can you pass the salt?").
(3) **cpc-name-correct:** A binary metric for evaluating if the new information or CPC has a groundable name. We implement this with string matching as the the action or concept name must match perfectly as it is essentially a keyword for the robot architecture. This metric is especially important if it is desirable for the linguistic parse to be executable in a robotic architecture. There may be many ways to stay "grab" or "get", but the underlying robotic architecture might only have a single function called "grasp" that must be executed if we want the robot to grab something. Thus, the parser must correctly align a speaker's choice of word with the underlying executable function.

(4) **spc-length-correct:** A binary metric for evaluating if the parser was able to identify the correct number of givens or SPCs. It is implemented as a check on the length of the list of givens. The purpose of this metric is to ensure a level of semantic alignment with the robotic architecture. Recall, that the SPC captures a set of presupposed relations and entity bindings and thus, to fully understand an utterance, the robot must understand what the speaker assumes the robot already knows.

(5) **spc-precision:** A numeric metric (between 0 and 1) for evaluating how many of the SPCs in the predicted list of SPCs are also present in the ground truth parse. SPC-precision and SPC-recall (below) are both standard machine learning metrics used to evaluate the accuracy of the set of SPCs.

(6) **spc-recall:** A numeric metric (between 0 and 1) for evaluating how many of the SPCs in the ground truth list of SPCs are also present in the predicted parse.

(7) **is-isomorphic:** A binary metric for evaluating whether the variables named in the CPC and SPC are positioned correctly. We implement this metric by first constructing a graph of each of the predicted and ground truth parses. The nodes are either CPC/SPC names as well as variable names. Directed edges establish how variables connect to the CPC and SPCs. Edge weights are assigned to label the position of the variable in the CPC or SPC. If the predicted and ground truth graphs are isomorphic (including edge weight constraints), then the variables are correctly positioned. The graph-based metrics of is-isomorphic and is-matched are both intended to capture the alignment of variable bindings between the CPC and SPC. For example, if we have an utterance "put the red cup on the table", we want to ensure that the variable name, say 'VAR0', that is assigned to the red cup in the CPC, also is the same variable name referencing the cup and the red property in the SPC. In this example, the CPC might be "puton(VAR0, VAR1)" and the SPCs might be "{cup(VAR0), red(VAR0), table(VAR1), DEFINITE(VAR0), DEFINITE(VAR1)}"

(8) **is-matched:** A binary metric for evaluating whether variables correctly match the corresponding CPC and SPC. We implement this using the same graph parse as was used for the is-isomorphic metric, with some additional checks for ensuring that each variable has the same set of successor nodes.

(9) **did-it-parse:** A binary metric (only for the translators) for evaluating whether the translated output is parseable by a symbolic parser.

3.3 Models

We used pre-trained Llama-2 [19, 20] 3B and 7B models publicly available and open-sourced via the Huggingface ecosystem. It is worthwhile noting that our focus in this paper is not on comparing different LLMs in and of themselves, but instead on comparing different strategies for integrating them into a cognitive architecture. As such, for our evaluation we selected two canonical and open-access LLMs of different sizes for comparing our integration strategies. We performed parameter efficient finetuning (PFET) [12] using 4bit QLoRA adapters [8], which optimizes finetuning by backpropagating gradients through a frozen, 4-bit quantized pretrained Llama-2 model into Low Rank Adapters. These models were trained and evaluated on NVIDIA RTX 4090 cards. It is important to note that this is an illustrative study showcasing the value of the metrics and our evaluation paradigm. As such, our evaluation of the broad (and growing) range of LLM models is certainly not exhaustive. We used the Llama-2 models as these were state-of-the-art at the time of the experiments and open source. It is also worthwhile to note that in robotic settings, especially in human-robot interactions, speed of performance is a relevant consideration. Although we, do not supply a time-to-process metric, the response time of our fine-tuned models was in the order of milliseconds. In contrast, some of our preliminary experiments showed that developing in-context models of these integration approaches using commercial (proprietary) models like GPT-4 are several orders of magnitude slower, and as such unsuitable for natural human-robot interaction. Thus, we did not include these models in the present study. However, as noted earlier, our approach does not preclude testing against these other models, experiments we intend to do in future work.

model-name	valid-json	intent-correct	cpc-name-correct	spc-length-correct	is-isomorphic	is-matched	spc-precision	spc-recall
llama23b	0.995561	0.964492	0.936085	0.964936	0.926764	0.869063	0.918331	0.918923
llama27b	0.999556	0.983577	0.952508	0.913893	0.893032	0.893475	0.873206	0.873650
llama23b-context	0.381269	0.993786	0.974257	0.296494	0.207279	0.286285	0.733393	0.882231
llama27b-context	0.999112	0.993786	0.977807	0.993786	0.979139	0.942743	0.957390	0.957686

Table 1. LLM-as-a-Parser

model-name	did-it-parse	valid-json	intent-correct	cpc-name-correct	spc-length-correct	is-isomorphic	is-matched	spc-precision	spc-recall
llama23b	0.549933	1.000000	0.539281	0.486906	0.539281	0.490901	0.447847	0.509691	0.511170
llama27b	0.644030	1.000000	0.626720	0.582779	0.626276	0.585442	0.529960	0.594171	0.598905
llama23b-context	0.643586	1.000000	0.626720	0.580559	0.626720	0.584110	0.553040	0.612517	0.609854
llama27b-context	0.741678	1.000000	0.736352	0.727474	0.736352	0.727918	0.703506	0.721113	0.722000

Table 2. LLM-as-a-Translator + CCG parser

While we do not provide benchmark latency or memory metrics, we note that LLaMA-2 3B models typically run within sub-second inference time on consumer-grade GPUs (e.g., RTX 4090), whereas 7B models are noticeably slower. We used parameter-efficient fine-tuning (QLoRA) with 4-bit quantization to reduce training footprint and improve deployment feasibility, particularly important for real-time human-robot interaction.

3.4 Results and Discussion

"Known" Data: Tables 1 and 2 show the performance of the parser approach and translator approach, respectively, for the "known" but held-out test data. In Table 1, We can observe that three out of the four models performed well across all the metrics. We observed a significant drop in the performance of Llama2-3B-Context model on the structural metrics associated with variable binding and on correctly identifying the "givens". That is, this model appeared to struggle to find referring expressions and correctly map them to known properties. One reason for this is that only about 38% of the outputs were valid JSON entries. In our experiments, we tasked GPT-4 with automatically fixing such cases, but we suspect that either many of these errors were not fixable or incorrectly fixed.

In Table 2, the performance significantly drops across all metrics[3]. These results suggest that the context information may have helped the models generate better "simple English". Even so, of all those outputs that parsed at all (about 50-70%) roughly only 50-70% of those parsed correctly.

Figure 3 shows the aggregate performance, comparing the parser approach to the translator approach. The parser outperforms the translator regardless of the presence of context information, but context appears to help the translator. Thus, our first hypothesis (that the translator will perform better than a parser) cannot be supported by the current experiment. In addition, our second hypothesis (that performance will improve with context), appears to track for smaller models like Llama-2-3B, but not so for larger Llama-2-7B model.

Style Variations: Figures 4 and 5 show how each model performs on each of our metrics across different input utterance styles. While there was very little by way of variations between models across the styles for any particular metric, we can observe a moderate drop in performance for all the models with respect to the correct variable binding ("is_matched") in the presence of indirect speech acts (see bottom left group of bars in Figure 4). It is worth noting that indirect speech acts constitute a large percentage of human imperative speech and correctly handling them is essential to smooth and normative human-robot communications.

[3]The perfect performance of valid-json metric is due to the fact that those that did parse, did so perfectly. But most outputs did not parse.

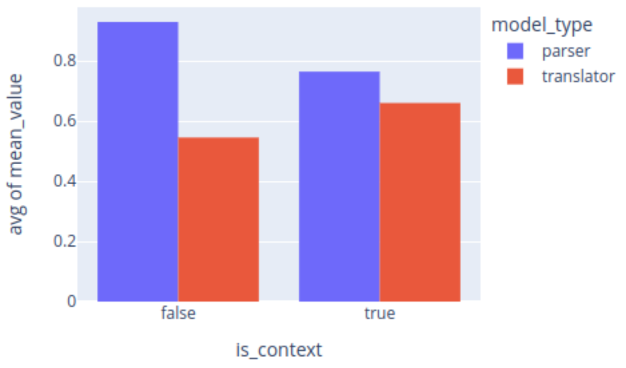

Fig. 3. Aggregated Performance (averaged over all metrics across both Llama-2 models) in the "known" data condition.

model-name	valid-json	intent-correct	cpc-name-correct	spc-length-correct	is-isomorphic	is-matched	spc-precision	spc-recall
llama23b	0.995552	0.978973	0.001213	0.528508	0.234533	0.000000	0.599137	0.891630
llama27b	0.999596	0.962394	0.016983	0.549535	0.227254	0.000404	0.621310	0.902952
llama23b-context	0.442378	0.986656	0.000809	0.029923	0.006066	0.000000	0.390315	0.811161
llama27b-context	1.000000	0.957946	0.228872	0.537404	0.224019	0.028306	0.597520	0.877342

Table 3. LLM-as-a-Parser (NOVELTY)

model-name	did-it-parse	valid-json	intent-correct	cpc-name-correct	spc-length-correct	is-isomorphic	is-matched	spc-precision	spc-recall
llama23b	0.417307	1.000000	0.413668	0.000000	0.272543	0.112818	0.000000	0.273487	0.364604
llama27b	0.532956	1.000000	0.531743	0.000000	0.329155	0.148403	0.000000	0.350452	0.476614
llama23b-context	0.188435	1.000000	0.180348	0.000000	0.127376	0.049333	0.000000	0.127241	0.154468
llama27b-context	0.350991	1.000000	0.350182	0.000000	0.263243	0.123736	0.000000	0.248955	0.303680

Table 4. LLM-as-a-Translator + CCG parser (NOVELTY)

"Novel" Data: Tables 3 and 4 show the performance of the parser approach and translator approach, respectively, for the "novel" but held-out test data. Across the board, we observe a significant drop in the performance of these models on the novel data as compared to the known data. Our hypothesis was that context information (i.e., action and property names) would help improve the performance of these models on novel data. However, as seen in Figure 6 context does not appear to impact the performance. Specifically, providing actions and properties within the prompt context does not appear to help the finetuned LLMs better parse or translate utterances. Additional experiments will need to be performed to explore the scope and limits of this preliminary finding.

A qualitative advantage of the parser-based integration is that its output maintains symbolic structure and intermediate representations (intent, CPC, SPC), which are interpretable and introspectable. This enables system designers to audit and debug the language pipeline step by step, a key benefit over opaque, end-to-end LLM approaches.

4 GENERAL DISCUSSION

Increasingly, robotic systems are gaining the ability to learn entirely new actions that are compositions of primitive actions during task performance and from humans directly. Once they have learned a novel action, robots will need to be able to use these newly learned actions in future requests. Our current results suggest that

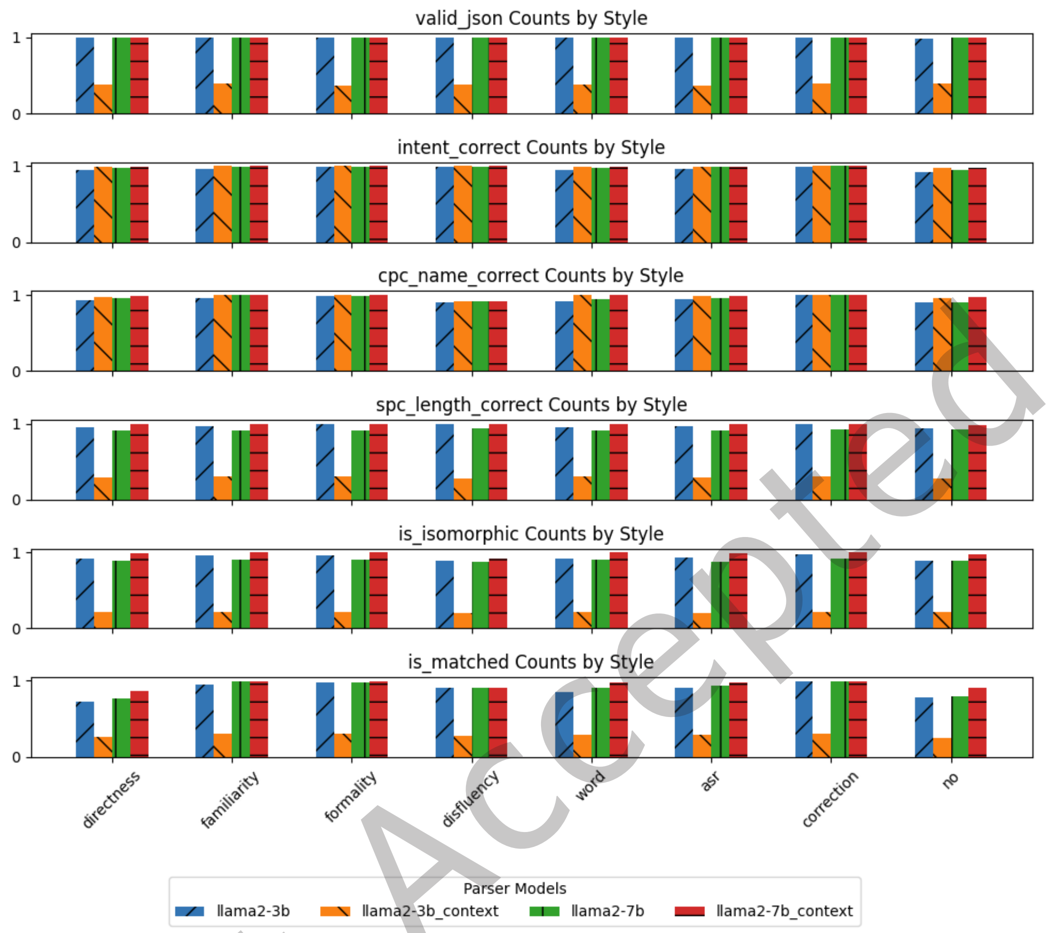

Fig. 4. Parser performance on various metrics across Different Styles

these capabilities are not readily realizable with existing LLMs. We do not suggest this is impossible. In fact, it may be that other architectural capabilities like constraint-guided decoding and retrieval augmented generation might be needed over and above the language model itself to improve its performance. We will explore these options in future work.

In this paper, our focus has been to compare two different LLM integration strategies against each other, rather than comparing one LLM with another. Although certain capabilities are enhanced with LLM size and scale, research has yet to show conclusively that future larger or smaller LLMs will perform as better or worse translators or parsers. Therefore, we do not currently have reason to believe that future larger/smaller LLMs will perform differently. For example, some research has shown that larger pretrained language models are not uniformly better than smaller models, as they can perform worse on a significant fraction of instances [34]. Moreover, all our models tested in this paper were finetuned with our data. One question is whether finetuning efficacy improves for translation and parsing tasks with increased model size. This is relevant to our results as improved translation and worsened parsing results could, in theory, flip our conclusions. However, we believe

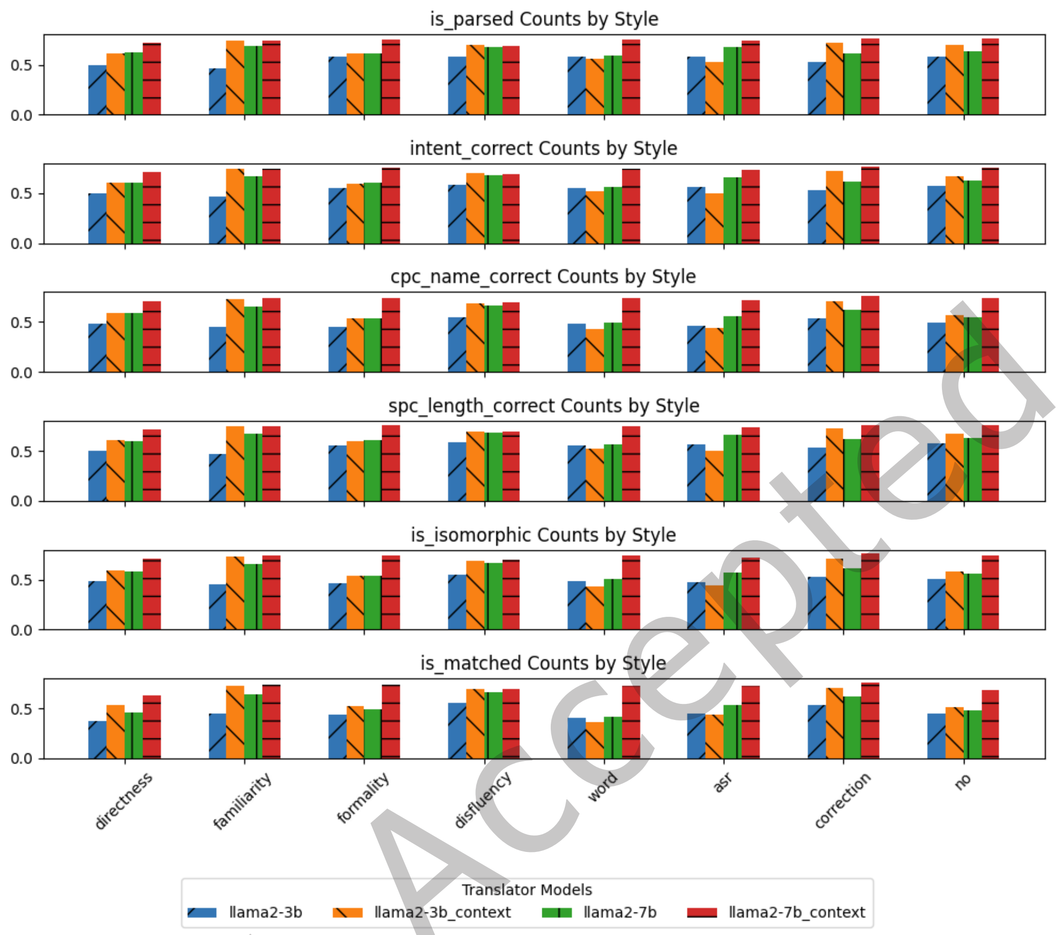

Fig. 5. Translator performance on various metrics across different styles

this is not likely to be the case, as finetuning efficacy is not just influenced by model size, but by other factors in the finetuning process. For example, research suggests that Parameter Efficient Finetuning (PEFT) (the approach we take in this paper) can improve performance without massive resources, potentially bridging the gap between smaller open-source models and larger LLMs [1]. Optimal finetuning method depends on the task and available data [33], and these findings suggest that the choice between different models (larger or smaller) is dependent on specific use cases and resource constraints. In a similar vein, it is unclear how the strategies will perform with models much smaller than 3B. Extrapolating from the current results, we hypothesize that such models are likely to be amenable to finetuning and as such are likely to maintain performance for “known” data but likely worse for “novel” data.

An important general question raised by the integration of LLMs into robotic system is the extent to which the gain in features (e.g., better and more comprehensive parsing, more general planning, better object recognition and scene understanding, etc.) trades-off with guarantees of behavior given that LLMs tend to confabulate even when they are finetuned and specifically trained to not do so. An LLM parser might thus increase the risk of

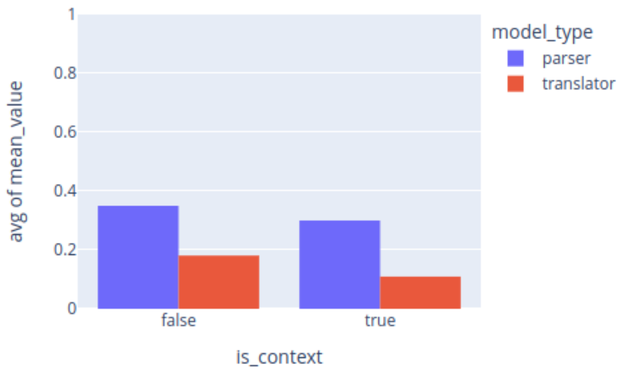

Fig. 6. Aggregated Performance in the "novel" data condition.

faulty behavior of the system compared to the behavior obtained with the more limited component that got replaced. While a thorough discussion of this trade-off and the various potential implications of integration LLMs into robotic architectures is beyond the scope of this paper – here we only compare different integration methods – it is worth pointing out that the systematic effects of any type of LLM integration need to be carefully considered and the pros and cons weighed against normative requirements and ethical principles.

Finally, since our focus has been on strategies for LLM integration into robotic platforms, it is worth considering questions about long-term sustainability and computational demands placed on these platforms. This is especially relevant given robotic systems require real-time performance in dynamic environments, and as such questions of latency and compute will play an important role. In this paper, we did not focus on this potential concern because there has been a substantial amount of ongoing research being done to integrate LLMs into robotics, and as such we expect questions of compute and latency to diminish as LLMs and also robotic architectures improve in their performance.

From a systems perspective, the choice of integration strategy should be guided by the existing architectural capabilities and the cost-benefit tradeoffs of modification. If a robot already employs a modular, component-based NL pipeline with components for parsing, grounding, and reasoning, integrating an LLM as a translator—i.e., adapting the LLM's output to match the expected symbolic representations—can be a practical and efficient option. This approach requires careful prompt design and potentially fine-tuning the LLM to produce structured output compatible with downstream components, as we have done in this study. However, if such a pipeline does not already exist, the engineering effort required to build or maintain a symbolic parser may outweigh the benefits. In such cases, replacing the entire NL subsystem with an LLM acting as a parser may offer a simpler path to deployment, especially when robustness to linguistic variability is critical. Our evaluation framework can help practitioners navigate these tradeoffs by assessing how different integration strategies affect performance, interpretability, and generalization.

A critical consideration for real-world robotics is safe execution. While LLM-based components improve flexibility, they introduce stochasticity and potential confabulation. One promising approach is a 'test-and-check' strategy, where LLM outputs are parsed and validated against logical or symbolic constraints before execution. Such verification layers are essential for safe, embodied deployment and warrant future study.

5 RELATED WORK

Traditionally, robotic architectures featured a Natural Language Processing (NLP) pipeline that extracts semantic and syntax information; for example, using Combinatory Categorial Grammar (CCG) [21, 23] that allows for the simultaneous extraction of syntax and semantics. These grammars can be used to connect action imperatives in the utterance with function calls in the robot. To overcome issues related to handling messy and ambiguous human speech, Large Language Models (LLMs) offer the promise of serving as a useful tool within the robotic architecture [28]. LLMs have been used for breaking down high-level instructions into a low-level executable plan of feasible actions [2, 27, 30], for code-generation within a robotic architecture [16, 24], for self-talk [13], tracking world state [31], detecting an analyzing failures [17] and summarizing multimodal input. In addition, there has been quite a bit of recent work in leveraging LLMs for planning [15, 22]. Research in the generation of formal language from natural language also shares many similarities to our proposed work, for example, in NL-to-SQL work [10] there is the notion of context (where the table schema is provided), a notion of question (similar to user utterance), an SQL query (which is similar to our lifted semantic parse). However, in addressing the informational considerations of human communication, current strategies for integrating LLMs with robotics fall short. As outlined by [3], these considerations involve the speaker's attention, the intended focus for the addressee, what is already known, priority of information, and background information. Failure to sufficiently address these areas may impede successful human-robot interaction which fundamentally depends on comprehending human communication as a diverse speech act with varied content and purposes.

6 CONCLUSION AND FUTURE WORK

Recent advances in LLMs have made it possible to use NL as *lingua franca* in robotic systems leading to an influx of new approaches for bringing the LLM's fluency, language understanding, commonsense knowledge, reasoning, and planning within an embodied robotic context. What has been missing is a high-level characterization of these different approaches and metrics for evaluating them. Here, we have proposed a set of metrics together with a suite of techniques to augment natural language data to evaluate the LLM-Robot integration across a range of syntactic, semantic, and pragmatic features that influence natural language understanding in embodied settings. We employ this framework and the metrics in probing a specific question of whether an LLM should be integrated into the NLU pipeline as a translator whose output can be passed through a symbolic parser making it more interpretable, or as a parser itself whose output can be directly used for action execution in the robot. Our results suggest that parsers outperform translators significantly, making the tradeoff between explainability and performance even more challenging. We also explored whether additional context about robot's action and perceptual repertoire can aid in handling novel actions but did not find evidence for the benefits of such an approach[4].

Our results suggest that LLM-as-Parser integration, while more effortful to engineer, provides substantially better semantic grounding and robustness to linguistic variation. These findings offer actionable guidance for system builders seeking a balance between interpretability and performance in natural language robotic systems.

Our current experiments only begin to scratch the surface of a fertile ground for research. More research needs to be conducted to expand these metrics and also study the LLM's ability to handle sets of objects, deictic speech, conjunctions, goal-based instructions, instructions that involve objects that are literals (e.g., "display Hello on your screen"), action plausibility based on known pre-/post conditions, learning instructions and procedural knowledge, extended dialog, and incremental speech processing.

[4]Code: We will make code repositories and video demos available at the time of publication.

REFERENCES

[1] Divyanshu Aggarwal, Ashutosh Sathe, and Sunayana Sitaram. 2024. Maple: Multilingual evaluation of parameter efficient finetuning of large language models. *arXiv preprint arXiv:2401.07598* (2024).

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. [n. d.]. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. ([n. d.]).

[3] Jennifer E. Arnold, Elsi Kaiser, Jason M. Kahn, and Lucy Kyoungsook Kim. 2013. Information Structure: Linguistic, Cognitive, and Processing Approaches. *Wiley interdisciplinary reviews. Cognitive science* 4, 4 (2013), 403–413. https://doi.org/10.1002/wcs.1234

[4] Chitta Baral, Juraj Dzifcak, Marcos Alvarez Gonzalez, and Jiayu Zhou. 2011. Using Inverse λ and Generalization to Translate English to Formal Languages. In *Proceedings of the Ninth International Conference on Computational Semantics*. Association for Computational Linguistics, 35–44.

[5] Joyce Chai, Qiaozi Gao, Lanbo She, Shaohua Yang, Sari Saba-Sadiya, and Guangyue Xu. 2018. *Language to Action: Towards Interactive Task Learning with Physical Agents.* 9 pages. https://doi.org/10.24963/ijcai.2018/1

[6] Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Wang, Zhiguo Wang, Vittorio Castelli, Bing Xiang, and Patrick Ng. 2023. DR.SPIDER: A DIAGNOSTIC EVALUATION BENCH- MARK TOWARDS TEXT-TO-SQL ROBUSTNESS. (2023).

[7] Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics* 33, 4 (2007), 493–552. https://doi.org/10.1162/coli.2007.33.4.493

[8] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. https://doi.org/10.48550/arXiv.2305.14314 arXiv:2305.14314 [cs]

[9] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn. 2009. What to Do and How to Do It: Translating Natural Language Directives into Temporal and Dynamic Logic Representation for Goal Management and Action Execution. In *2009 IEEE International Conference on Robotics and Automation*. IEEE, Kobe, 4163–4168. https://doi.org/10.1109/ROBOT.2009.5152776

[10] Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving Text-to-SQL Evaluation Methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 351–360. https://doi.org/10.18653/v1/P18-1033 arXiv:1806.09029 [cs]

[11] Jeanette K. Gundel, Nancy Hedberg, and Ron Zacharski. 1993. Cognitive Status and the Form of Referring Expressions in Discourse. *Language* 69, 2 (1993), 274–307. https://doi.org/10.2307/416535 jstor:416535

[12] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. https://doi.org/10.48550/arXiv.2304.01933 arXiv:2304.01933 [cs]

[13] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022. Inner Monologue: Embodied Reasoning through Planning with Language Models. arXiv:2207.05608 [cs]

[14] Pat Langley, John E. Laird, and Seth Rogers. 2009. Cognitive Architectures: Research Issues and Challenges. *Cognitive Systems Research* 10, 2 (June 2009), 141–160. https://doi.org/10.1016/j.cogsys.2006.07.004

[15] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. arXiv:2304.11477 [cs]

[16] Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. 2023. Lang2LTL: Translating Natural Language Commands to Temporal Robot Task Specification. arXiv:2302.11649 [cs]

[17] Zeyi Liu, Arpit Bahety, and Shuran Song. 2023. REFLECT: Summarizing Robot Experiences for Failure Explanation and Correction. arXiv:2306.15724 [cs]

[18] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. 2009. ROS: An Open-Source Robot Operating System. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*. Kobe, Japan.

[19] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code Llama: Open Foundation Models for Code. arXiv:2308.12950 [cs]

[20] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong,

Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code Llama: Open Foundation Models for Code. https://doi.org/10.48550/arXiv.2308.12950 arXiv:2308.12950 [cs]

[21] Matthias Scheutz, Tom Williams, Evan Krause, Bradley Oosterveld, Vasanth Sarathy, and Tyler Frasca. 2019. An Overview of the Distributed Integrated Cognition Affect and Reflection DIARC Architecture. In *Intelligent Systems, Control and Automation: Science and Engineering*. 165–193. https://doi.org/10.1007/978-3-319-97550-4_11

[22] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2022. ProgPrompt: Generating Situated Robot Task Plans Using Large Language Models. (2022). https://doi.org/10.48550/ARXIV.2209.11302

[23] Mark Steedman. 2001. *The Syntactic Process*. MIT Press.

[24] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. [n. d.]. ChatGPT for Robotics: Design Principles and Model Abilities. ([n. d.]).

[25] Nguyen Vo, Arindam Mitra, and Chitta Baral. 2015. The NL2KR Platform for Building Natural Language Translation Systems. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Vol. 1. 899–908.

[26] Bailin Wang, Zi Wang, Xuezhi Wang, Yuan Cao, Rif A. Saurous, and Yoon Kim. 2023. Grammar Prompting for Domain-Specific Language Generation with Large Language Models. arXiv:2305.19234 [cs]

[27] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An Open-Ended Embodied Agent with Large Language Models. https://doi.org/10.48550/arXiv.2305.16291 arXiv:2305.16291 [cs]

[28] Lilian Weng. 2023. LLM Powered Autonomous Agents. https://lilianweng.github.io/posts/2023-06-23-agent/.

[29] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. 2016. Fetch and Freight: Standard Platforms for Service Robot Applications. In *Workshop on Autonomous Mobile Service Robots*. 1–6.

[30] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. 2023. TidyBot: Personalized Robot Assistance with Large Language Models. https://doi.org/10.48550/arXiv.2305.05658 arXiv:2305.05658 [cs]

[31] Takuma Yoneda, Jiading Fang, Peng Li, Huanyu Zhang, Tianchong Jiang, Shengjie Lin, Ben Picker, David Yunis, Hongyuan Mei, and Matthew R. Walter. 2023. Statler: State-Maintaining Language Models for Embodied Reasoning. arXiv:2306.17840 [cs]

[32] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2019. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. arXiv:1809.08887 [cs]

[33] Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2024. When scaling meets llm finetuning: The effect of data, model and finetuning method. *arXiv preprint arXiv:2402.17193* (2024).

[34] Ruiqi Zhong, Dhruba Ghosh, Dan Klein, and Jacob Steinhardt. 2021. Are Larger Pretrained Language Models Uniformly Better? Comparing Performance at the Instance Level. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. 3813–3827.

Received 29 February 2024; revised 27 May 2025; accepted 19 June 2025