Instructional Text Across Disciplines: A Survey of Representations, Downstream Tasks, and Open Challenges Toward Capable AI Agents

Abdulfattah Safa[1,2*], Tamta Kapanadze[2], Arda Uzunoğlu[3], and Gözde Gül Şahin[1,2,4]

[1]KUIS AI Lab, Koç University, Istanbul, Türkiye
https://gglab-ku.github.io/
[2]Koç University, Istanbul, Türkiye
asafa22@ku.edu.tr
[3]Johns Hopkins University, Baltimore, Maryland, USA
[4]Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany

Recent advances in large language models have demonstrated promising capabilities in following simple instructions through instruction tuning. However, real-world tasks often involve complex, multi-step instructions that remain challenging for current NLP systems. Robust understanding of such instructions is essential for deploying LLMs as general-purpose agents that can be programmed in natural language to perform complex, real-world tasks across domains like robotics, business automation, and interactive systems. Despite growing interest in this area, there is a lack of a comprehensive survey that systematically analyzes the landscape of complex instruction understanding and processing. Through a systematic review of the literature, we analyze available resources, representation schemes, and downstream tasks related to instructional text. Our study examines 181 papers, identifying trends, challenges, and opportunities in this emerging field. We provide AI/NLP researchers with essential background knowledge and a unified view of various approaches to complex instruction understanding, bridging gaps between different research directions and highlighting future research opportunities.

* Corresponding author.

Action Editor: Yuki Arase. Submission received: 27 June 2025; revised version received: 23 December 2025; accepted for publication: 24 February 2026.

https://doi.org/10.1162/COLI.a.616

1. Introduction

The ability to program machines/computers with natural language, if it can be made successful, would fundamentally change the relationship between humans and computers. As of today, only a small percentage of humans (less than 1%) have the necessary skill set to program their computers or phones to perform new tasks. Machines and computers, however, are mostly viewed as preprogrammed devices with a fixed-set of skills. To move towards that goal, *programmable machines—more recently coined as a new term: AI agents*—[1] should be equipped with (at least) excellent instruction understanding capabilities.

With the recent advances in the Natural Language Processing (NLP) field, Large Language Models (LLMs) have demonstrated capacity on understanding instructions (Naveed et al. 2024). This is mostly achieved via "instruction tuning" that performs supervised fine-tuning on base language models with a large amount of instruction-response pairs (Zhang et al. 2023c). These pairs are typically single sentences, describing a low-level task that can be performed in a single step. More recently, instruction-tuned models are further trained with human preferences and reasoning traces (Guo et al. 2025) (e.g., chain-of-thought) via various reinforcement learning techniques (Rafailov et al. 2023; Ouyang et al. 2022) that boost their performance on standard reasoning benchmarks (Hendrycks et al. 2021a; Rein et al. 2024; Hendrycks et al. 2021b; Li et al. 2022d). This has ultimately led to another research direction: LLM-based AI agents that can interact with environments such as Web sites (Deng et al. 2023; Lai et al. 2024; Zhou et al. 2023; He et al. 2024a) using tools such as Python interpreters (Wang et al. 2024b; Yang et al. 2023; Schick et al. 2023) to autonomously pursue user goals. Even though modern LLM-based agents show promise in Web navigation and tool use, they still struggle with complex, multi-step instructions[2] which contain temporal (Gonzalez-Pumariega et al. 2025), conditional (Shridhar et al. 2020b; Ghazarian et al. 2025), hierarchical dependencies (Zhou et al. 2023; Dagan, Keller, and Lascarides 2025).

Furthermore, the ability to "follow instructions" of such models is mostly evaluated on simple scenarios with simple instructions (He et al. 2024b) (e.g., a single event with an explicit set of arguments), on simulated environments that approximate real-world complexity. However, real-world tasks require resolving event and argument ambiguities in underspecified instructions; as well as understanding and following more complex instructions, which are mostly multi-step directives containing temporal, conditional, hierarchical dependencies. Hence, the next frontier in NLP—in particular, LLM-based agents—research will be to understand these complex instructions in a real-world-like setting.

Understanding such instructions requires—at minimum—understanding of events, the relations between events, their participants and the environment, and be able to perform multi-hop, common sense reasoning with such event knowledge. Various fields have investigated related subtopics, primarily as computational linguistics and NLP (e.g., event semantics, semantic parsing, script/scenario generation, common-sense

1 Historically, programmable machines have been used to define systems that can be configured through natural language rather than code. Recently, it has evolved into today's AI agent paradigm, where users *prompt* LLM models with natural language to pursue their goals in an environment using external tools such as Python interpreter.

2 State-of-the-art LLM agents achieve 8–30% success rates (Xu et al. 2025), requiring an average of 27–40 interaction steps to completion depending on the model, with performance particularly degraded on long-horizon tasks requiring multi-step reasoning across changing contexts.

reasoning, LLM agents), robotics (e.g., manipulation via instructions), business intelligence (e.g., process models), and computer vision (e.g., recipe understanding). Different fields use different naming conventions for complex instructions (e.g., process, procedure, task), different techniques to represent them (graph, workflow, business model, programming language, etc.) and different venues to publish, which creates a high barrier to entry for new researchers.

The main goal of this survey is to equip the AI—in particular NLP—researchers with the necessary background knowledge and provide guidance on the future challenges and opportunities for conducting research on complex instructions. In contrast to existing surveys on tangential areas (see §2), we provide a holistic presentation on *available resources* and *the range of tasks* related to instructional text, making connections to other fields such as robotics, business intelligence, and computer vision. With this survey, we aim to answer the following research questions:

- **RQ1:** What are the most *common ways to represent* long-form, a.k.a., multistep, *instructional text* across different disciplines? What corpus/corpora (raw or structured) are available for various representation schemes?
- **RQ2:** Which *downstream tasks* are readily available on instructional text? How do they differ by means of domain, methods, and evaluation metrics?
- **RQ3:** What *recurring challenges* persist across tasks despite methodological advances, and what do these patterns reveal about fundamental gaps in current approaches?

In Section 2, we identify the related areas and existing surveys that complement this work. Next, in Section 3, we define our PRISMA-based survey methodology (Page et al. 2021a) and present the bibliographic characteristics of the reviewed papers. Sections 4 and 5 aim to answer **RQ1** and **RQ2**, respectively. We identify the common themes and remaining challenges separately for each task in Section 5 to answer **RQ3**. We present a taxonomy for the data representation types, and the range of tasks to guide the reader in Figure 1.

2. Related Work

The goal of this survey is to provide an interdisciplinary—in particular, NLP, robotics, business intelligence, and computer vision—view on instructional text research. In other words, it focuses on (i) how different fields represent instructions; (e.g., unstructured, event-centric, entity-centric and symbolic); (ii) how instructional text is used across fields (e.g., entity-centric data $\rightarrow$ entity state tracking, symbolic data $\rightarrow$ robotic navigation); and (iii) identifying recurring challenges and outlining a roadmap for future research. Due to the large number of data representation formalisms and associated downstream tasks surveyed, there are many adjacent fields that might intersect with the surveyed content such as semantic parsing. This work does not serve as a comprehensive guide on each of the adjacent fields (e.g., another survey on semantic parsing), but rather a guide that talks about those fields *in the context of multi-step, complex instructional*

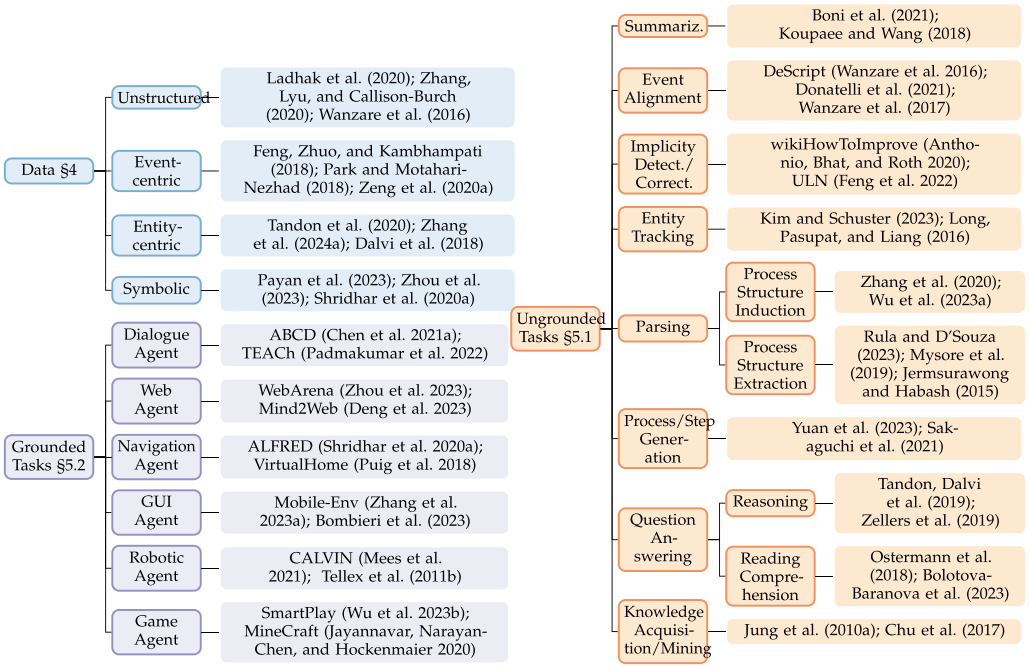


Figure 1
Taxonomy of instructional text research by means of data representation schemes and downstream tasks. **Data** representations include Unstructured, Event-centric, Entity-centric and Symbolic formats. **Tasks** are split into two main categories: Grounded (Dialogue/Web/Navigation/GUI/Robotic/Game Agents) and Ungrounded (Summarization, Event Alignment, Implicit Instruction Detection/Correction, Entity Tracking, Parsing, and Question Answering).

text (e.g., instruction parsing). We identify the two most related fields to procedural language understanding as:[3] (i) event understanding, and (ii) grounding.

Event Understanding. Event-centric NLP field primarily deals with extracting event information from textual documents. Xiang and Wang (2019) compile various approaches and challenges for event extraction from textual data, providing an overview of tasks, methods, and performance metrics. The survey categorizes techniques ranging from pattern matching to advanced machine learning models. Chen et al. (2021b) offer a comprehensive tutorial[4] on event-centric information extraction, prediction, and knowledge acquisition, focusing on event-centric tasks and methodologies. In addition, Li et al. (2022b) survey the deep learning techniques for event extraction, exploring sophisticated models that detect, categorize, and analyze events across various domains. **Semantic parsing**, a subfield of event understanding, aims to map natural language to logical forms or database queries. The subfield has a rich history and a large number of dedicated surveys (Kamath and Das 2018; Li, Qu, and Haffari 2020; Chen et al. 2025)

3 We use the terms *complex instructions*, *procedure*, *process*, and *script* interchangeably to refer to sequences of natural language directives guiding task completion; terminology varies by field (NLP: instructions/procedures; cognitive science: scripts; business intelligence: processes).
4 Cognitive Computation Group.

focusing on semantic formalisms (e.g., abstract meaning representation) and parsing methodology. Unlike these surveys, we focus on the higher-level relations between events, participants, and their environments in long procedural text rather than extracting atomic event information.

Grounding Instructions. In broad terms, grounding generally refers to a type of task that involves connecting language to some form of external knowledge or real-world context such as images, knowledge bases, robot arms, and even operating systems. Chandu et al. (2021) discuss the evolution of the term "grounding" and draw connections to cognitive science. More recently, several surveys have been published on specific environments and grounding types. For instance, Cohen et al. (2024) survey different meaning representations for grounding robotic language for navigation/manipulation tasks, while Wang et al. (2024a) discuss various aspects of LLM agents on grounded tasks. In contrast, our objective is to survey the wide range of applications and environments to ground complex instructions rather than focusing on one environment or approach. Finally, one literature relevant to grounded tasks is code generation, i.e., program synthesis. Here, similar to Web or mobile environments, the goal is to generate code that can be executed by a symbolic interpreter (e.g., Python, SQL) on a certain environment. However, the focus is learning the conversion process in an offline manner through large annotated sets, rather than through environment interaction (Payan et al. 2023). We refer readers to comprehensive surveys on program synthesis (Jiang et al. 2024; Zan et al. 2023).

To the best of our knowledge, there is no survey focusing on procedural text. The only resource is the tutorial by Zhang (2022) that compiles a set of selected resources containing procedures, along with a set of selected applications. In contrast, we provide a *systematic* methodology and taxonomy covering a considerably wider range of representation types, resources, and tasks for procedural text. We also extend the scope to other fields such as robotics, business intelligence, and computer vision, aiming to provide a unified perspective across disciplines. In addition, our survey is related to **scripts**—a sequence of events with multiple actors—and **planning**, i.e., generating a feasible (and hopefully minimal) sequence of steps to achieve a specific goal. Although we mention them where related (e.g., script generation, Web agents), we consider them to be outside the scope of this survey, and refer the readers to the classical book by Schank and Abelson (2013). Finally, the boundary between the discussed paradigms is not absolute. For instance, work on semantic parsing for robot commands (Tellex et al. 2011a; Matuszek et al. 2012) bridges both generating executable symbolic representations while learning from environmental feedback.

3. Methodology

We have followed the guidelines given in the PRISMA statement (Page et al. 2021a, 2021b) to conduct this systematic review. The overview of our methodology is given in Figure 2. It contains four steps: (i) identification, (ii) screening, (iii) eligibility check, and (iv) inclusion, which are explained below.

3.1 Identification

We selected various libraries to ensure broad coverage of publications in fields relevant to this survey, such as AI and its subfields—particularly NLP, robotics, machine learning, and industrial engineering. DBLP indexes key computer science works, while

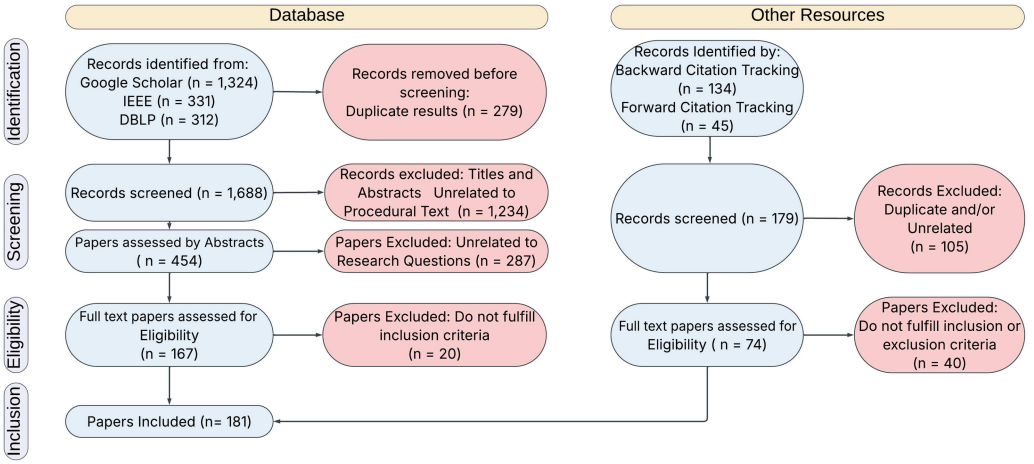


Figure 2
Study protocol.

IEEE Xplore provides a wider range of scientific content. Scholar primarily indexes peer-reviewed papers without field distinction. Given the rapid publication rate in NLP, we also considered notable non-peer-reviewed ArXiv papers. The review period spans 2010 to the present to capture relevant publications. We used the Google Scholar API from Serp API for our search, manually extracting other digital library entries. Because Google Scholar API lacks abstract retrieval, we sourced missing abstracts from Semantic Scholar. The full list of queried keywords that are based on research questions to capture relevant papers while maintaining a manageable volume are given in Appendix A.

Figure 3 shows the distribution of retrieved publications across different keywords and databases after deduplication. An exact phrase match (EPM) searches for the exact phrase in papers, while exact word match (EWM) matches individual words. EPM is supported by Google Scholar and IEEE Xplore, whereas DBLP supports only EWM. To account for variations in terms (e.g., "instruction" vs. "instructions"), we searched for both forms and combined the results. The AND operator was used to combine search terms for Google Scholar and IEEE Xplore, while DBLP was searched for each word individually with the AND operator. From all digital libraries, we retrieved up to a hundred search results per keyword returned by the database, as papers beyond were highly irrelevant. The execution of search queries in the four digital libraries returned 1,967 papers. After deduplicating the merged results from the databases, 1,688 papers remained. As expected, Google Scholar and IEEE Xplore return more results since they index a broader range of venues and domains, while DBLP only indexes computer science papers. We observe that the most number of papers are retrieved via the keyword "natural language instructions" since it covers various applications and methodologies. The least number of papers are retrieved for more specialized, narrow fields such as "entity state tracking" and "recipe parsing".

3.2 Screening

Our screening process involved two phases. First, we manually reviewed titles and abstracts to remove irrelevant papers, starting with an initial pool of 1,688 papers. During this inspection, we noticed that papers relevant to our **RQ1** and **RQ2** often

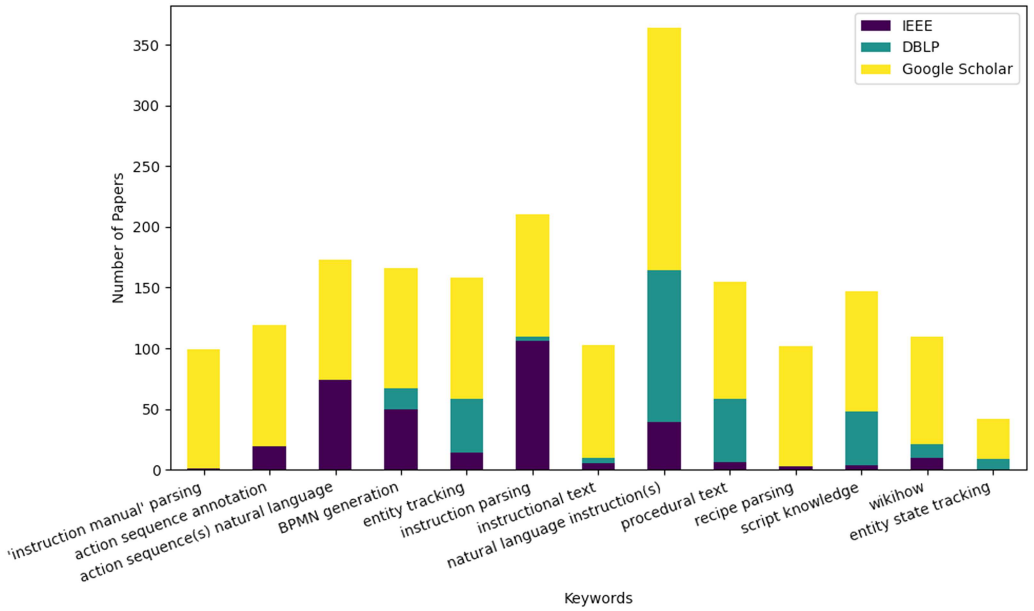


Figure 3
Publications per keyword per database.

included keywords like "dataset," "evaluation set," "benchmark," or "test set." This makes sense, as the studies of interest typically introduce new datasets on instructional text or novel downstream tasks using existing resources, or sometimes both. These papers were then further filtered based on these keywords, reducing the count to 454.

3.3 Eligibility

Assessing the eligibility entailed manual inspection. Assigned researchers carefully examined the abstracts of the filtered papers to determine their alignment with our research inquiries. This manual review significantly narrowed down the selection to 167 papers.

3.4 Inclusion

We curated a shared library of relevant papers for comprehensive full-text review, refining the selection to 147 papers. After retrieving the papers from digital libraries, we performed forward and backward citation tracking, identifying 34 additional papers, resulting in a final count of 181 papers. Once the list was finalized, we tagged the papers based on tasks, data representations, and research locations. This tagging helped us develop a taxonomy of representation types and downstream tasks (related to **RQ1** and **RQ2**) and visualize the bibliographic properties of the publications.

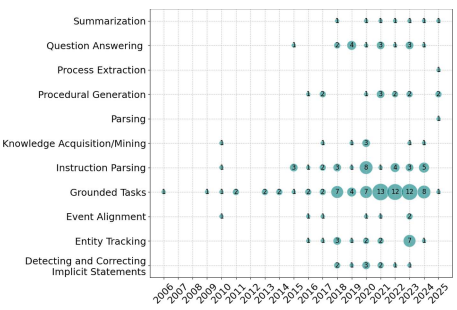
3.5 Bibliographic Properties

To gain a comprehensive view of the instructional text processing research landscape, we visualized various data aspects. Figure 4 shows the geographical distribution of

Figure 4
Geographical distribution of publications.

publications, with a large number coming from renowned research hubs like Stanford, Seattle, New York City, and Beijing. This highlights the concentration of research efforts in urban centers known for their academic institutions. Notably, Stanford and Seattle lead in publication numbers, reflecting strong research output in NLP, especially in procedural text fields. This dominance, also seen in other AI subfields (Breit et al. 2023), underscores the lack of diversity in languages and research groups.

Figure 5a shows the temporal distribution of tasks, illustrating a steady growth in *Summarization* and *Question Answering*, with a notable peak in the latter starting from 2018. Newer fields such as *Grounded Tasks* have seen a significant rise in interest since 2017, reflecting a shift towards more contextual and task-driven AI applications. The



(a) Papers Per Task Distribution over Time

(b) Papers from each country Per Task Distribution over Time

Figure 5
Comparison of task distribution and country distribution over time.

increasing trend in *Instruction Parsing* from 2018 onward indicates renewed interest in more structured task interpretation. Meanwhile, *Procedural Generation* has seen a consistent, albeit slower, development. Traditional areas such as *Entity Tracking* and *Event Alignment* have remained relatively steady, while areas like *Knowledge Acquisition/Mining* have gained more prominence in recent years. The growth from 2018 onwards, particularly in these emerging areas, suggests a trend toward more dynamic and interactive AI systems.

Figures 4 and 5b illustrate the global scope of instructional text research, with certain countries excelling in specific areas. The USA pioneered Step Inference in 2006 and leads in *Grounded Tasks* and *Instruction Parsing*. Germany has focused on *Instruction Parsing* since 2016. China, active since 2018, has expanded into *Instruction Parsing* and *Entity Tracking*. Japan and India contribute to *Summarization* and *Procedural Generation*. Key research hubs include San Francisco, Berlin, and Beijing. These findings suggest potential collaborations, such as the USA and Germany on Parsing tasks or China and India on domain-specific applications, to advance global instructional text research. It should be noted that the geographic and temporal distributions are influenced by our choice of databases (Google Scholar, IEEE Xplore, DBLP) and the dominance of English as the primary language in AI research. Despite our best efforts, our selection criteria may not comprehensively capture work from under-represented communities who might publish elsewhere (e.g., local venues for cost efficiency) in their local language; and under-represented fields such as process extraction in business intelligence, where research might appear in smaller venues using varying terminologies.

4. Data

Even though a substantial amount of work for procedural text relies on **unstructured** corpora derived from Web sources like WikiHow,[5] the field is not short of structured, labeled datasets. We categorize structured representations into three main categories: (i) **event-centric** (see §4.2), (ii) **entity-centric** (see §4.3), and (iii) **symbolic** (see §4.4). Event-centric representations mostly ignore the properties of entities, and how certain events affect those. The entities are mostly referred to as "the argument" of the event—a.k.a., action—and are represented as free text. Instead, they put the events/tasks into focus and define a rich set of relations between them (subsequent, conditional, concurrent, parent/child, etc.). On the other hand, entity-centric representations predefine intrinsic properties of entities (e.g., color, location) and track the state changes caused by extrinsic and intrinsic events. The events are mostly not tied to any static knowledge base or lexicon and are simply represented as free text. Finally, symbolic representations aim to transform the natural language instructions into a machine-readable format, such as first-order logic, robot actions, and UI actions, so that they can be executed/performed on a specific simulated environment. They are mostly used for grounding tasks (see §5).

Most **event-centric** representation schemas are linguistically motivated. For instance, SIMMR (Jermsurawong and Habash 2015) define a dependency grammar to represent action-entity relations in recipes. Similarly, a large number of studies (e.g., Feng, Zhuo, and Kambhampati 2018; Mysore et al. 2019) employ semantic representations inspired from common semantic formalisms (e.g., PropBank, FrameNet) with

5 https://www.wikihow.com/.

varying levels of granularity. Although most entity-centric representations avoid linguistic formalisms, one exception (Clark et al. 2018) leverages Verb Network (VerbNet) as an additional background knowledge for effect (i.e., postcondition) prediction. While linguistically motivated frameworks provide foundational tools and inspiration for researchers in the field, procedural text often requires domain-specific adaptations or extensions—such as fine-grained domain-specific arguments (Mysore et al. 2019), exclusive arguments (Feng, Zhuo, and Kambhampati 2018), concurrent event relations (Sakaguchi et al. 2021; Jermsurawong and Habash 2015), and implicit entity handling Kiddon et al. (2015). Notions such as exclusive arguments, richer event-event relations, and implicit entities are not fully handled by traditional semantic formalisms; and are an active area of research.

4.1 Unstructured

The field has mainly used three techniques to create unstructured corpora to study procedural text: (a) scraping/filtering available Web resources, (b) crowdsourcing, and (c) synthetic data generation.

4.1.1 Web Corpora. The most popular way to construct large-scale corpora that can be repurposed (or annotated) for many downstream tasks has been scraping Web sites that contain how-to information. WikiHow has been exploited tremendously to study procedural, a.k.a., instructional text. Even though it is referred to as "unstructured", WikiHow articles follow strict writing style, possessing a certain file structure. For instance, each article contains a goal *(e.g., How to destroy an old computer?)*, several methods to reach the goal *(e.g., destroying a computer entirely, destroying a computer for recycling)*, and steps to execute the method *(1. Wipe your hard drive, 2. Remove battery, etc.)*. Furthermore, each article has a community Q&A and warning/tips sections for people following the article to seek help. WikiHow has a considerably high domain coverage—from make-up tips to writing indie songs. Articles are curated and edited by experts from all around the world in different languages. Nonetheless, similar to other Web corpora, it is *predominantly in English*. Furthermore, most articles contain pictures that accompany the step description, which makes WikiHow suitable for multimodal studies.

In summary, WikiHow has been the most popular resource due to (i) the strict style followed by the authors that enabled easy scraping, (ii) high domain coverage (19 domains, and subdomains), (iii) large-scale (e.g., more than 235,000 articles), (iv) multilinguality, (v) multimodality, and (vi) high quality (i.e., expert curated articles). In addition to WikiHow, Instructables, ifixit, and ehow, several recipe Web sites such as Allrecipes, Cookpad, HaoDou, Food, and Xiachufang have been used (Yagcioglu et al. 2018; Nabizadeh, Kolossa, and Heckmann 2020; Liu et al. 2018; Zhang, Yamakata, and Tajima 2022; Diwan, Batra, and Bagler 2020; Jung et al. 2010; Zhang, Yamakata, and Tajima 2021).

Web corpora are used in many shapes and forms, i.e., for pretraining or fine-tuning language models to enhance reasoning or planning capacities. It is also repurposed for a variety of downstream tasks, such as summarization (Ladhak et al. 2020), question answering (Zhang, Lyu, and Callison-Burch 2020; Zellers et al. 2019; Yang et al. 2021), classification (Zhou, Shah, and Schockaert 2019; Park and Motahari-Nezhad 2018), detecting implicit information in instructions (Anthonio, Bhat, and Roth 2020), learning task hierarchy (Zhou et al. 2022), generation (e.g., next step, all steps) (Lyu, Zhang, and Callison-Burch 2021; Nguyen et al. 2017), and benchmarking procedural language

understanding and planning abilities (Uzunoglu and Sahin 2023; Uzunoglu, Şahin, and Safa 2024) without any additional annotation layers. Even though recipes and WikiHow are mostly exploited, we find that *many resources such as troubleshooting Web sites from tech companies* (e.g., Canon *are overlooked* in the literature with some exceptions (Goyal et al. 2021). The reasons might be i) their small size ii) the lack of a consistent structure that makes them hard to parse. Another *overlooked resource is local Web sites* (e.g., Turkish food recipes), which might be due to the lack of enough NLP experts or interest in local communities.

4.1.2 Crowdsourcing. Researchers have used *crowdsourcing to create smaller, but more targeted datasets*. For instance, DeScript (Wanzare et al. 2016) is a small corpus of scenarios on everyday tasks such as going shopping, riding a bus, getting a haircut, and taking a bath. It contains a rather limited number of tasks (only 40), however, each one has been described step-by-step by 100 different crowdtaskers. Steps, a.k.a., event sequence descriptions, written by different taskers are later aligned with each other. Several other studies, such as Task2Dial (Strathearn and Gkatzia 2022) and ABCD (Chen et al. 2021), use crowdworkers to generate dialogue datasets grounded on instructional documents such as recipes and call center guidelines by assigning different roles to crowdworkers (e.g., call center employee, information giver on a certain recipe). It is also commonly used to annotate existing small corpora for a specific task. Such tasks are mostly related to extracting some form of information from instructional text, e.g., tools from fixing manuals (Nabizadeh, Kolossa, and Heckmann 2020), ingredients from recipes (Zhang, Yamakata, and Tajima 2022). Due to the costs associated with crowdsourcing, this technique has been mostly used for generating scripts, grounded dialogues, or to add small annotation layers to existing corpora. We find that, even though, for example, the corpus DeScript (Wanzare et al. 2016) and the alignments are publicly available, *such resources are overlooked and not used for related downstream tasks, e.g., event paraphrasing or alignment*, to the best of our knowledge, with some exceptions (Wanzare et al. 2017).

4.1.3 Synthetic Data Generation. Synthetic datasets and environments are also common for creating toy setups under simplifying assumptions. For example, Weston et al. (2016) formulate 20 question answering tasks to evaluate different linguistic and reasoning abilities such as co-reference resolution and temporal and spatial reasoning. The stories are generated in a simulated world which is defined by manually written rules (e.g., *one should find food if hungry*) on a predefined set of entities with predefined attributes such as location and size. Similarly, Kim and Schuster (2023) define a set of entities (book, hat, etc.), properties (e.g., location) and a small set of actions (e.g., MOVE) to generate synthetic procedures about moving entities between different boxes via a simple Python script. It is then used to evaluate the entity state tracking abilities of large language models. Another research line develops instruction following game environments such as TextWorld (Marc-Alex et al. 2018) as a testbed for reinforcement learning agents to measure their planning and exploration abilities. Similar to others, the environment is defined via a fixed set of rooms, entities, actions, and entity properties; however, the game is interactive and played step-by-step.

Synthetic generation gives researchers the ability to *control* the complexity of the task and the environment. However, synthetic datasets mostly *lack the linguistic diversity, and long-tail, rare events that cannot be programmed*. In other research areas of NLP, it is common to post-process the synthetic data via asking crowdsourcers to *paraphrase the synthetic text (Kim and Schuster 2023) to increase linguistic diversity. However, we don't observe the same trend for procedural text literature, which could be a promising future direction*.

Similar to Web corpora, synthetic data is also commonly used for pretraining models. For instance, Shi et al. (2022) create a large-scale synthetic dataset by randomly sampling the environment states and associated programs to pretrain a BART-large model, then evaluate on entity state tracking test splits.

4.1.4 Generation with LLMs. One of the latest trends in the field is to *exploit large language models to generate silver, large-scale datasets*. Instructional text literature also *follows this trend*. For instance, Yuan et al. (2023) create a constrained planning dataset (CoScript) of 55,000 procedures, e.g., procedures to make a cake for *diabetics*, by first generating with InstructGPT and then filtering with constraints.

4.2 Event-Centric Representation

Despite the differences in naming conventions, we use event, a.k.a., instruction, to refer to a step in a procedure that contains an action phrase to accomplish a subgoal (e.g., *Bake in the oven*). We define action verb as the predicate (e.g., Bake), and the arguments of the predicate (e.g., it = the thing that is baked, oven = Location) as the entities. In Table 1, we demonstrate the diversity of representations from different angles, on a set of work that is manually selected to give the best overview. We determine the differences to be along the formats used for the *actions* and *entities*, and the *entity roles w.r.t the action*, and *event-event relations*.

We note that, *business information literature* that uses Business Process and Model Notation (BPMN) *uses a different terminology*. Here, an event is represented as an empty circle to mark the start, middle, and end of the process; while the event is called an action or an activity. Examples of event-centric representations, such as "baking a cake" and the BPMN, can be found in Appendix E.

4.2.1 Actions and Entities. We use event, action, and task interchangeably following the previous work since the *granularity of the task representations varies greatly*. They refer to *a single step in the instruction* that needs to be performed to accomplish a goal. Furthermore, the majority of the work makes a simplifying assumption that a step/task

Table 1
Common task-centric representation schemes for procedural text. OD: Open domain, i.e., no domain-specific tags. Ex: Exclusive, Es: Essential, Opt: Optional, Suc: Successive, Con: Concurrent, PC: Parent-Child, Pre: Precondition, Eff: Effect.

	Action	Entity	Action-Entity	Event-Event	OD
Wiki HG (Feng, Zhuo, and Kambhampati 2018)	Text	Text	Ex, Es, Op	Suc, Ex	✓
MS (Mysore et al. 2019)	Text	Tagged	Tagged	Suc	✗
SIMMR (Jermsurawong and Habash 2015)	Tagged	Tagged	Input, Output	Suc, Con	✗
Action Graph (Kiddon et al. 2015)	Text+Implicit	Text, Food, Location	✗	✗	✗
APSI (Zhang et al. 2020)	Text	✗	✗	Suc	✓
proScript (Sakaguchi et al. 2021)	Text	✗	✗	Suc, Con	✓
InScript (Modi et al. 2016)	Tagged	Tagged	✗	Suc	✗
Process Graph (Park and Motahari-Nezhad 2018)	Text	Text	Actor, Time, Location	Suc, Opt, Con, PC, If-Else	✓
Process Model (Qian et al. 2020)	Text	Text	Object	Suc, Opt, Con	✓
BPMN (Zeng et al. 2020)	Text	✗	Object	Suc, Ex, Con	✓

contains only one action. For instance, Sakaguchi et al. (2021) take the *full step* (e.g., *bake for the right amount of time*) as the task representation, while Zhang et al. (2020) represent them as an *action phrase* that contains only one predicate and argument (e.g., *search car, apply loan*). Another line of work (Park and Motahari-Nezhad 2018; Qian et al. 2020; Zeng et al. 2020) uses the BPMN (Kocbek et al. 2015)—an industry standard—and similarly *represents a task as a predicate-argument pair*, where the argument is always an object. All the studies mentioned above use a free text representation (denoted as "Text" in the Action and Entity columns of Table 1), sometimes limiting the number of actions to the most frequent ones. *This assumption might be problematic, especially when the articles contain different author styles* as in WikiHow, or DeScript. To address that, *several studies limit the domain and use domain-specific tags*. For instance, Modi et al. (2016) define *scenario-specific events* (e.g., apply-soap, undress, turn water on) and tag all the action phrases. Similarly, SIMMR (Jermsurawong and Habash 2015) use a *predefined recipe-specific language* named MILK to represent the actions and ingredients. These are noted with "Tagged" in the Action and Entity columns of Table 1.

The studies that use free text representation mostly annotate the indices of the objects of interest (action verb, action phrase, argument, etc.) (Feng, Zhuo, and Kambhampati 2018; Mysore et al. 2019), or simply extract them with existing tools like semantic role labelers (Zhang et al. 2020; Park and Motahari-Nezhad 2018; Qian et al. 2020). *The majority presume that all actions are explicitly stated in the text*.

Although there is a growing body of work on implicit instructions (see §5.1.3), *event-centric representations mostly ignore implicit entities*. For instance, imagine the phrase *"Mix a, b, c and then bake"*. The thing that is baked is the implicit, unmentioned mixture. Mysore et al. (2019) explicitly acknowledge such cases—"a material entirely absent from the text being the true argument"—but exclude them from their annotation schema, as their framework does not allow arguments to have multiple parents when entity states change. This design choice, while pragmatic, results in systematic exclusion of semantically crucial information necessary for tracking ingredient transformations through procedural chains.

Entities are marked (e.g., for the position) in the text (Feng, Zhuo, and Kambhampati 2018; Park and Motahari-Nezhad 2018; Qian et al. 2020), tagged with predefined entity types (Jermsurawong and Habash 2015; Modi et al. 2016; Mysore et al. 2019), or simply ignored (Sakaguchi et al. 2021; Qian et al. 2020). Kiddon et al. (2015) extract entities, however, only label a few important entity types, namely, as food and location. Jermsurawong and Habash (2015) represent the ingredients/entities and actions of the recipe as terminal and internal nodes, respectively.

4.2.2 Action-Entity Relation. The role the entity plays in the action is crucial for task representation. Several studies (Mysore et al. 2019; Modi et al. 2016) define highly fine-grained entity types that also include information about the relation (e.g., Material_of). *Studies on scripts (Zeng et al. 2020; Sakaguchi et al. 2021; Modi et al. 2016) mostly ignore such relations, while others do not have any consensus*. For instance, Feng, Zhuo, and Kambhampati (2018) define the arguments as exclusive, essential, and optional. Imagine the phrase "Take a pen or a pencil". Here, pen and pencil are defined as exclusive arguments of action "take". Essential and optional arguments are also common in semantic schemas (e.g., Arg-0, Arg-TMP in PropBank). However the "actor" role rarely exists in procedural text, while the text is mostly imperative. Furthermore, *exclusive arguments are not part of the standard semantic schemes (e.g., PropBank, FrameNet, VerbNet), however, is important for procedural text*.

Table 2
Examples of event relations from the surveyed literature.

Relation	Event A	Event B	Explanation	Example Paper
Exclusive	Pay by cash	Pay by credit	Either/Or relation	Wiki-HG (Feng, Zhuo, and Kambhampati 2018)
Successive	Receive feedback	Process feedback	Event B happens after Event A	MS (Mysore et al. 2019)
Concurrent	Book a flight	Book a hotel	Event A and Event B occur simultaneously	SIMMR (Jermsurawong and Habash 2015)
Optional	Book a flight	Add travel insurance	Event B is optional	Process Model (Qian et al. 2020)
Parent Child	Write a paper	Do literature search	Event B is the child process of Event A	Process Graph (Park and Motahari-Nezhad 2018)
Conditional	Application is received	Application is confirmed	There is a condition for moving from Event A to Event B, e.g., GRE>80	proScript (Sakaguchi et al. 2021)

Despite the representation power and ample choices in BPMN, the *surveyed studies mostly focus on the "object", ignoring other essential information such as duration and location* with some exceptions (Park and Motahari-Nezhad 2018). Another approach is to represent such relations with unlabeled dependency links (Jermsurawong and Habash 2015). Here, the entities are given as input to the action. Unlike others, this enables linking entities with multiple actions, representing *implicit* entities such as mixtures with a link towards the root and representing entities and actions on the same level.

4.2.3 Event-Event Relation. One important feature that *differentiates the field of procedural language understanding from event extraction is the rich set of higher-level relations between tasks/events*. *Defining procedures with only successive events has the minimal representation power, but has been the default way*. This simplistic approach is problematic and is mostly addressed by business information management, and software engineering literature via defining and standardizing richer relations such as "exclusive", "successive", "concurrent", and "optional" (see Table 2 for examples). Some exceptions in the NLP field are Feng, Zhuo, and Kambhampati (2018) and Sakaguchi et al. (2021), including exclusive, and concurrent relations, respectively. Another exception is Jermsurawong and Habash (2015), whose work can also handle concurrent events implicitly in a dependency tree structure.

The parent-child relation for procedures is also understudied. The reasons are that procedures are mostly studied in isolation due to their challenging structure, and links to subtasks are usually nonexistent. Similarly, *conditional tasks are vastly ignored*. To study those, one needs alternative methods and the conditions to switch. Even though WikiHow contains different methods to accomplish a goal, which one to choose is not explicitly stated, and inferred by the readers. *Although there are exceptions, hierarchy and conditional processes are severely understudied according to our survey results*.

4.3 Entity-Centric Representation

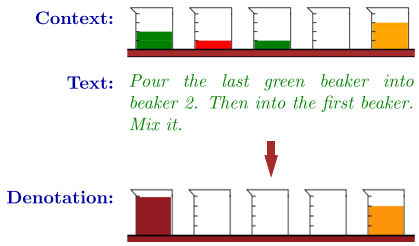
In Table 3, we compile the common entity tracking studies and analyze them in several dimensions, such as domain coverage and the complexity (e.g., number of steps, number of actions).

Table 3
Common entity tracking datasets. OD: Open domain.

	Entity Types	# States	# Actions	# Entities	Avg # Steps	# Examples	OD
ProPara v1.0 (Dalvi et al. 2018)	Multiple	2	3	<10	>4	488 paragraphs	✗
OpenPI (Tandon et al. 2020)	Unlimited	Unlimited	Unlimited	Unlimited	n.a	810 articles, 30K state changes	✓
NPN (Bosselut et al. 2018)	Multiple	6	384	Varying	n.a	65K recipes	✗
Boxes (Kim and Schuster 2023)	Multiple	2	3	Varying	12	2200 scenarios	✗
SCONE (Long, Pasupat, and Liang 2016)	1	<3	<6	<6	5	4K scenarios	✗

The datasets fall into two main categories: *simulated* and *crowdsourced*. Boxes (Kim and Schuster 2023), TextWorld (Li, Nye, and Andreas 2021), and SCONE (Long, Pasupat, and Liang 2016) use simulation to easily keep track of object states and can be easily extended. For the simulated ones, the vocabulary is generally small—approximately 1,100 to 1,200 words, and the text is artificially generated, hence usually uses a simple generative grammar and might contain spurious correlations. Kim and Schuster (2023) attempt to increase the language diversity by including paraphrases. The second category of datasets include ProPara v1.0 (Dalvi et al. 2018), OpenPI (Tandon et al. 2020) and NPN (Bosselut et al. 2018); and use natural language text written by experts where the entities and states are annotated by crowdworkers. Opposite to simulated text, natural procedures are linguistically rich. Thus, the same entity types or states can be referred to using different names. Recently released OpenPI-v2.0 (Zhang et al. 2024) attempts to address this by clustering similar objects (e.g., spice and seasoning) together for a more uniform dataset. Figure 6 shows two examples from simulated (left) and crowdsourced (right) datasets.

One important difference between the two approaches is the initial environment/world state. Simulated datasets explicitly include *statements about the initial state* of all entities (e.g., Box 1 is empty) *while real-world procedures/processes do not* mention



(a) An example from the ALCHEMY dataset. The content of each beaker at each step is visualized showing the results of the corresponding action.

		Participants:				
Paragraph (seq. of steps):		water	light	CO_2	mixture	sugar
	state0	soil	sun	?	-	-
Roots absorb water from soil						
	state1	roots	sun	?	-	-
The water flows to the leaf.						
	state2	leaf	sun	?	-	-
Light from the sun and CO_2 enter the leaf.						
	state3	leaf	leaf	leaf	-	-
The light, water, and CO_2 combine into a mixture.						
	state4	-	-	-	leaf	-
Mixture forms sugar.						
	state5	-	-	-	-	leaf

Time

(b) An example from ProPara dataset. Each filled row shows the existence and location of participants between each step.

Figure 6
Examples of Entity Tracking Scenarios from the Alchemy (Long, Pasupat, and Liang 2016) (simulated) and ProPara (Dalvi et al. 2018) (crowdsourced) datasets.

the initial states but are required to be inferred. For instance, for the photosynthesis process, we assume (but it is not explicitly stated) that the plants are buried in the ground, they have leaves, and there is sun above, etc. There are several studies (Clark et al. 2018; Zhang et al. 2021b) that aim to *inject* such common sense knowledge through external knowledge bases. For instance, (Clark et al. 2018) use Stanford Research Institute Problem Solver (STRIPS)-style Semantic Lexicons to define preconditions and effects of actions. They use commonsense persistence algorithms to propagate states when information is missing. For some of the domains such as cooking (e.g., Onion: NotCooked, Pan: Empty), default initial states can be easier to identify when the task is restricted to a closed vocabulary of fixed attributes, such as existence and location (e.g., ProPara [Dalvi et al. 2018]), while not so easy for other expert domains, e.g., technical manuals (Goyal et al. 2021). However, the difficulty increases significantly in open-vocabulary datasets (e.g., OPENPI [Tandon et al. 2020]) where entities, attributes, and state values are unrestricted. On the other hand, *the goal is known beforehand in crowdsourced work* (e.g., how to fix Wifi adapter, produce energy by photosynthesis), although the *end states are again not explicitly stated but can be inferred* (e.g., Adapter: fixed). Meanwhile, *simulated work is mostly random simulations without any clear goal*, i.e., things moving from box to box for no reason. But the *final states are again well-defined* because they are deterministically computed based on the explicit actions and initial state definitions in the closed world model. Another important difference is the "validity" of the scenarios. *Crowdsourced* procedures written by humans are *valid* by nature; however the simulated environments need to have certain measures for the validity of the generated scenario. For instance Li, Nye, and Andreas (2021) define the game as valid if it is possible to reach the end goal from the initial state.

Because these studies focus on tracking the state changes of the entities, the number of entity types and possible states are domain-specific and limited to a small number. One exception is OpenPI (Tandon et al. 2020) and its derivatives (Zhang et al. 2024; Wu, Li, and Ji 2023) that consider all entity and state configurations in hundreds of WikiHow articles from various domains. The earliest work by Long, Pasupat, and Liang (2016) define only a single entity type such as a beaker, bottle, or a person. Later studies (Bosselut et al. 2018; Dalvi et al. 2018) increase the number of entity types, i.e., Bosselut et al. (2018) keep a short list of ingredients, where Dalvi et al. (2018) focus on a small set of entities that are part of physical processes (e.g., photosynthesis) and Li, Nye, and Andreas (2021) incorporate text adventure game objects (e.g., player, chest).

For the sake of tractability, the number of possible actions that can be applied to entities are limited (from 2 to 384). Most actions are generic—not domain-dependent—such as create, remove, move; while NPN (Bosselut et al. 2018) also includes several domain-specific actions (e.g., bake, boil, cut). Entity properties, a.k.a. states, are again defined as a small set of fixed states (between 1 and 6). Although they are mostly domain-independent (e.g., Location, Existence), some are domain-specific (e.g., Cookedness). States can be binary (edible, isOpen, exists, etc.), categorical (shape, color), or free text (location). Note that location for simulated datasets are dominantly categorical (Box1, Beaker1, etc.).

Another factor that affects the complexity of the task is the number of steps that cause at least one state change in the set of entities. For both the simulated and crowdsourced datasets, the researchers use a threshold for the minimum and maximum number of steps. For instance, Dalvi et al. (2018) sets the minimum to 4, while the simulated ones (Kim and Schuster 2023; Long, Pasupat, and Liang 2016) explicitly set the number of steps to 12 and 5, accordingly. In theory, the simulated datasets can be easily extended, however, currently they are restricted to a few thousand scenarios

(2K–4K). On the other hand, crowdsourced datasets are generally smaller, mostly containing fewer than 1,000 articles. Finally, several datasets provide additional annotations for supporting tasks. For instance, Dalvi et al. (2018) provide an additional dependency explanation graph to include a cause-effect relation between the sentences; and OpenPIv2.0 (Zhang et al. 2024) introduces an additional "saliency score", i.e., importance in accomplishing the task for entities in the range of 1–5.

4.4 Symbolic Representation

Entity- and event-centric representations capture procedural structure but lack executability—the ability to directly drive automated agents in interactive environments. Executable symbolic representations address this limitation through formal languages (e.g., PDDL, LTL, domain-specific action languages) that can be automatically interpreted and executed in simulation environments. These representations enable grounding natural language instructions to concrete environment actions, though this executability introduces distinct design constraints absent in purely descriptive formalisms.

This final category of work generally defines—or uses a predefined—formalism/language that can be automatically interpreted by an external symbolic interpreter and executed/performed on a simulation environment interactively. The type of symbolic language depends on the environment in which it is interpreted. To name a few, Planning Domain Definition Language (PDDL) (Miglani and Yorke-Smith 2020), Linear Temporal Logic (LTL) (Bombieri et al. 2023), and First-Order-Language (FOL) (Long, Pasupat, and Liang 2016) are commonly used in robotic environments for navigation and manipulation tasks. UI actions related to mobile and operating systems are used for environments such as AndroidHowTo (Li et al. 2020), PixelHelp (Li et al. 2020), and WinHelp (Branavan et al. 2009). Custom-made, simplistic languages are designed mostly for 3D simulation and game environments like ALFRED (Shridhar et al. 2020a), VirtualHome (Puig et al. 2018), JerichoWorld (Ammanabrolu and Riedl 2021), FAVOR (Li et al. 2024), TextWorld (Marc-Alex et al. 2018), and SmartPlay (Wu et al. 2023b). In addition to environments focused on UI actions and operating systems, the procedural text is also used in the realm of code generation Payan et al. (2023). We discuss the grounded tasks and surveyed environments in more detail in §5.2.

Symbolic formalisms reflect a *recurring tension*: The more you can express, the harder the system becomes to learn and use. General-purpose planning languages like PDDL maximize expressiveness (Miglani and Yorke-Smith 2020). They support first-order logic, conditional effects, and complex precondition–effect specifications, typically represented as ⟨Name, Parameters, Preconditions, Effects⟩ tuples. **Temporal logics** like LTL take a different approach, emphasizing temporal structure through operators such as next and until—constructs that map naturally onto how we describe procedures in language (Bombieri et al. 2023).

On the other hand, *domain-specific* languages sacrifice this expressiveness for practical gains. By restricting the action vocabulary to small primitive sets, i.e., 12 household actions, or even just 4 UI commands, these formalisms become easier to learn and more robust (Puig et al. 2018; Branavan et al. 2009; Li et al. 2020). Yet even with a limited scope, they maintain compositional structure, separating actions from arguments (`pour(beaker1, beaker2)`, `click(OK_button)`). This compositionality enables systematic grounding between linguistic expressions and environment entities (Miglani and Yorke-Smith 2020; Bombieri et al. 2023; Puig et al. 2018; Li et al. 2020).

Design Constraints. Moving from description to execution introduces design constraints. To translate natural language like utterance "make coffee" into executable steps, first the system needs to infer that one must walk to the coffee maker before pressing its button—*commonsense prerequisites* that natural language leaves implicit (Puig et al. 2018). Second, *completeness* proves even trickier. When researchers analyzed human-authored programs, they found only 64% were actually complete (Puig et al. 2018). Another 28% skipped minor steps (sitting down before standing up), while 8% omitted crucial actions entirely. Two approaches have emerged for handling this: **Prescriptive** approaches simply restrict what people can say, allowing only predetermined verb forms and connectors (Bombieri et al. 2023). **Accommodative approaches** take the opposite tack: Accept linguistic variation but normalize it aggressively through lemmatization and co-reference resolution (Miglani and Yorke-Smith 2020). Neither approach is perfect. Symbolic structures demand token-level precision, yet procedural knowledge is inherently flexible, e.g., *walk to the kitchen, go to the kitchen, and head to the kitchen* all mean the same thing (Puig et al. 2018; Miglani and Yorke-Smith 2020).

Data Creation. Creating symbolic training data introduces practical challenges. First, *semantic equivalence*: Multiple symbolic representations can express identical behavior. While parsers with lemmatization can normalize synonyms (mapping "walk" and "go" to the same action [Miglani and Yorke-Smith 2020]), structurally different programs can achieve identical goals (Puig et al. 2018; Payan et al. 2023). VirtualHome found that the activity "Make coffee" had 69 functionally correct programs with only 26% structural overlap (Puig et al. 2018). This means that annotations must validate functional correctness rather than exact symbolic matches (Miglani and Yorke-Smith 2020; Payan et al. 2023). Validating functional correctness requires reference models or simulator execution, which is resource-intensive (Payan et al. 2023). Second, **linguistic naturalness**: Existing datasets have largely circumvented the functional correctness validation problem through synthetic generation, producing 295K templated commands in one study and 5,193 procedural programs in another (Li et al. 2020; Puig et al. 2018). While this reduces annotation costs, it produces linguistically constrained data: VirtualHome's synthetic programs exhibit vocabulary bias toward code-like expressions (Puig et al. 2018), and broader studies of synthetic text report decreased lexical and syntactic diversity compared to human-written data (Guo et al. 2024; Chen et al. 2024)

Cross-Domain Transfer. Symbolic formalisms require domain-specific infrastructure that hinders transfer. PDDL domains specify an ontology (predicates and objects) and action definitions with parameters, preconditions, and effects (Miglani and Yorke-Smith 2020). For instance, a PDDL action definition is a four-tuple `(Name, Parameters, Preconditions, Effects)` where preconditions are predicate values that must be met before execution (Miglani and Yorke-Smith 2020). LTL templates encode temporal-causal flows using logical operators, mapping natural language connectors (if/otherwise, then/once, until/repeat) to formal logic (Bombieri et al. 2023). Each formalism demands distinct annotation infrastructure. PDDL requires defining domain predicates and grounding them to commonsense ontologies (Miglani and Yorke-Smith 2020). LTL needs language constraint definitions specifying how temporal sequences and conditions are expressed in domain texts (Bombieri et al. 2023). Custom action primitives for simulators require separate predefined action lists (Puig et al. 2018; Wu et al. 2023b). Consequently, annotation efforts developed for one formalism rarely transfer to another (Miglani and Yorke-Smith 2020; Bombieri et al. 2023; Li et al. 2020). This fragmentation substantially increases the cost of scaling symbolic approaches across domains.

These challenges point toward two research directions we explore later. The expressiveness-learnability trade-off and difficulty ensuring functional correctness motivate improved data quality approaches (§6.8). The fragmentation of annotation infrastructure across formalisms motivates better neural-symbolic integration (§6.7).

4.5 From Representation to Task

The choice of data representation fundamentally shapes which downstream tasks can be addressed. In short, **symbolic representations** concentrate on grounded tasks because execution requires translating language into interpretable action primitives—a capability only symbolic formalisms can provide. **Entity-centric representations** specialize in entity state tracking, as they are mostly designed around predefined entity types, properties, and valid state values that enable precise evaluation while constraining the cross-domain transfer.

Event-Centric Representations. These dominate process structure extraction and induction tasks since they explicitly encode the relational structure (sequential, concurrent, conditional) that extraction and induction tasks aim to recover. **Unstructured corpora** serve as a generalist foundation, enabling massive scale without annotation burden but providing no explicit supervision for structured predictions. Figure 7 summarizes these representation–task associations and highlights underexplored pairings.

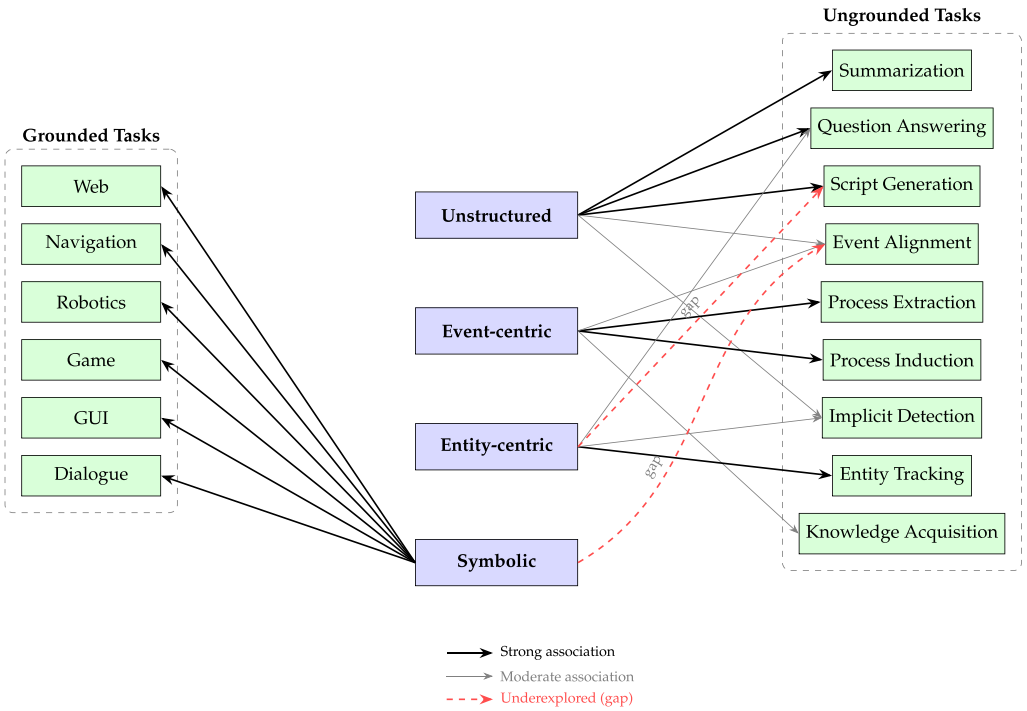


Figure 7
Representation-task associations in procedural language understanding. Strong edges indicate dominant pairings in the literature; moderate edges show secondary associations. Dashed red edges highlight underexplored combinations representing research opportunities: entity-centric representations for script generation tasks, and symbolic methods for alignment.

Two notable gaps emerge from this mapping. First, event-centric and entity-centric representations capture complementary aspects—action sequences versus state changes—yet remain largely separate. Real-world procedural understanding requires both: An agent following cooking instructions must know the action sequence *and* track ingredient states. Second, certain pairings remain unexplored, such as entity-centric approaches for generation or symbolic methods for alignment.

Beyond these theoretical gaps, practical constraints also shape the landscape. Representation-task associations manifest differently across application domains, as domain characteristics often dictate which representations are practical and which tasks are feasible. We provide a more detailed analysis below:

Recipes and Cooking Instructions Domains. These dominate the literature due to generally well-formed Web text mostly created by experts, abundant Web data (e.g., Recipe1M (Marin et al. 2019), WikiHow-sourced datasets and corpora, AllRecipes), and accessible semantics requiring no specialized expertise. These domains favor unstructured, event-centric and later, entity-centric data representations and support the broadest task diversity—from summarization and question answering to grounded execution—making cooking the *defacto* testbed for procedural understanding.

Scientific and Technical Domains (e.g., materials synthesis [Mysore et al. 2019], technical troubleshooting [Goyal et al. 2021]). These present a contrasting profile. These domains involve specialized lexicons—domain-specific action vocabularies and entity types such as chemical compounds and hardware components. Because of this specialization, annotation requires domain expertise, leading to high costs that constrain dataset scale. This pushes research toward event-centric representations with controlled vocabularies and extraction tasks.

Game and Simulation Domains (e.g., TextWorld [Marc-Alex et al. 2018], ALFRED [Shridhar et al. 2020a], Minecraft [Narayan-Chen, Jayannavar, and Hockenmaier 2019]). These use symbolic representations by design, as environments require executable action primitives. Event granularity varies from atomic UI actions to high-level goals, but linguistic diversity is typically sacrificed for environment compatibility. The tight coupling between representation and execution means these domains focus almost exclusively on grounded tasks.

Business Process Domains. These occupy a unique niche, using standardized formalisms like BPMN that encode rich event-event relations (successive, concurrent, conditional, exclusive, optional, parent-child) rarely captured in other domains (Park and Motahari-Nezhad 2018; Qian et al. 2020). This representational expressiveness enables sophisticated process extraction and induction but limits cross-domain transfer due to formalism-specific assumptions.

This domain concentration has methodological implications: The field's heavy reliance on cooking data means that models are optimized for and evaluated on procedures with accessible vocabulary, explicit action-object structure, and entity-centric state changes. Whether these models generalize to technical procedures, where implicit domain knowledge, specialized terminology, and different event granularities are the norm, remains largely untested. We return to these gaps and their implications for evaluation in §6. With these representations, task, and domain relationships established, we now examine each task category in detail.

5. Tasks

Plenty of downstream tasks have been proposed to evaluate procedural language understanding in general. These tasks are naturally constrained by the data representation. We broadly categorize the procedural tasks into two categories: (1) grounded and (2) ungrounded. **Grounded** tasks refer to the tasks where the instruction is grounded on an external database, document, or environment, whereas **ungrounded** tasks are solely defined on textual data. Please refer to Figure 1 for the taxonomy and to Appendix C for the full list of tasks and papers.

5.1 Ungrounded Tasks

In this section, we discuss various ungrounded tasks related to instructional text. Table 4 summarizes the landscape of ungrounded tasks, highlighting the datasets, models, and evaluation metrics that characterize each area.

Table 4
Summary of major ungrounded tasks in procedural text, highlighting commonly used datasets, modeling approaches, and evaluation metrics.

Task	Common Datasets	Common Models	Metrics
Summarization	WikiHow (Koupaee and Wang 2018) HowSumm (Boni et al. 2021)	T5 mBART BART Pegasus	ROUGE BLEU BERTScore METEOR
Event Alignment	Recipe1M (Marin et al. 2019) ARA Corupus (Donatelli et al. 2021) DeScript (Wanzare et al. 2016)	BERT LSTM HMM	Accuracy P/R/F1
Implicit Detection	R2R-ULN (Feng et al. 2022) KIDSCOOK (Bisk et al. 2019)	Pretrained LMs Vision-LMs	P/R/F1 Accuracy
Entity Tracking	ProPara (Dalvi et al. 2018) OpenPI (Tandon et al. 2020)	Neural Process Networks BERT variants	Accuracy P/R/F1 BLEU ROUGE
Process Structure Extraction	RecipeDB (Batra et al. 2020) Recipe1M (Salvador et al. 2017) MSComplexTasks (Zhang et al. 2021a) MIAIS (Zhang, Yamakata, and Tajima 2022)	Bi-LSTM CNN BERT GPT-4	P/R/F1 Success Rate BERTScore
Process Structure Induction	MiniWoB (Liu et al. 2018) DeScript (Wanzare et al. 2016)	MSA Clustering Seq2Seq GPT-4	E-ROUGE P/R
Procedural Generation	proScript (Sakaguchi et al. 2021) CoScript (Yuan et al. 2023) InScript (Modi et al. 2016) WIKIPLAN (Lu et al. 2023)	T5 GPT-3 LSTM/GRU	BLEU ROUGE BERTScore METEOR
Question Answering	WikiHowQA (Bolotova-Baranova et al. 2023) MCScript2.0 (Ostermann, Roth, and Pinkal 2019) HellaSwag (Zellers et al. 2019) RecipeQA (Yagcioglu et al. 2018)	BERT BART RoBERTa Triplet Networks	Accuracy ROUGE BLEU Recall@k
Knowledge Acq./Mining	Data sourced from WikiHow and Instruction manuals	BERT LongFormer	Accuracy Recall

5.1.1 Summarization. The trend focuses on extracting key instructions from either *single* (Ladhak et al. 2020; Koupaee and Wang 2018; Le and Tuan 2024) or *multiple* (Boni et al. 2021) procedural documents. This can be **extractive**, where key sentences are directly compiled from the source text, or **abstractive**, where new phrases are generated to summarize the content. WikiHow (Koupaee and Wang 2018) and HowSumm (Boni et al. 2021), discussed in 4.1, are the main resources. For instance, Koupaee and Wang (2018) generate human-crafted summaries for entire WikiHow articles, while Boni et al. (2021) produce extractive summaries for external documents. Le and Tuan (2024) generate extractive summaries from WikiHow without explicit labels, selecting key sentences that approximate their abstractive summaries. DeChant and Bauer (2022) utilize the ALFRED (Shridhar et al. 2020a) dataset to generate abstractive summaries of robotic actions, leveraging video frames and detailed action descriptions to produce both brief, one-sentence summaries and detailed, step-by-step instructions for each task. Methods have evolved from seq2seq models (Koupaee and Wang 2018) to fine-tuned Transformers such as T5, mBART, BART, and Pegasus (Ladhak et al. 2020; Boni et al. 2021; DeChant and Bauer 2022; Le and Tuan 2024; Kwon and Lee 2025), with ROUGE as the dominant evaluation metric, supplemented by BLEU, BERTScore, and METEOR (DeChant and Bauer 2022; Le and Tuan 2024; Koupaee and Wang 2018).

Limitations and Open Questions. The core limitation is the **optimization-task mismatch**: Models optimize sentence-level salience but are evaluated on summary-level coherence (Kwon and Lee 2025). This explains why systems underperform despite strong sentence selection: Independently chosen sentences fail to form coherent procedural sequences. We believe three deficiencies might drive this: (1) models arrange sentences by probability rather than semantic dependencies, causing step-ordering failures; (2) procedural details like measurements are lost when optimizing for salience (DeChant and Bauer 2022); and (3) metrics like ROUGE-L cannot penalize ordering violations (Kwon and Lee 2025), masking these failures. These evaluation limitations connect to broader metric inadequacy issues discussed in §5.3, with potential solutions in §6.3.

5.1.2 Event Alignment. Event alignment typically refers to the process of identifying and matching corresponding actions or steps across different sets of unstructured instructions. An example on recipes (Lin et al. 2020) is given in Figure 8. Event alignment is either one-to-one (Nandy, Kapadnis et al. 2023; Regneri et al. 2010; Wanzare et al. 2017), a.k.a. **paraphrasing**, or one-to-many (Lin et al. 2020; Donatelli et al. 2021).

The task is rich in domains thanks to the availability of raw procedural text. For example, several works (Nandy, Kapadnis et al. 2023; Lin et al. 2020; Donatelli et al. 2021) focus on the alignment of instructions within different cooking *recipes*—Nandy, Kapadnis et al. (2023) use a dataset containing 1 million recipes from Recipe1M+ (Marin et al. 2019), RecipeNLG (Bień et al. 2020), and various how-to blog Web sites, while Lin et al. (2020) introduce a dataset comprising 150K instruction pairwise alignments across 4,262 dishes. More recently, Donatelli et al. (2021) provide the Aligned Recipe Actions (ARA) corpus comprising recipes for 10 different dishes, originally sourced from Lin et al. (2020). Apart from recipes, a considerable amount of work (Regneri et al. 2010; Wanzare et al. 2016, 2017) focuses on *everyday tasks* and compile novel resources for Event Sequence Description (ESD) such as DeSCRIPT, SMILE, and OMICS.[6]

6 http://openmind.hri-us.com/.

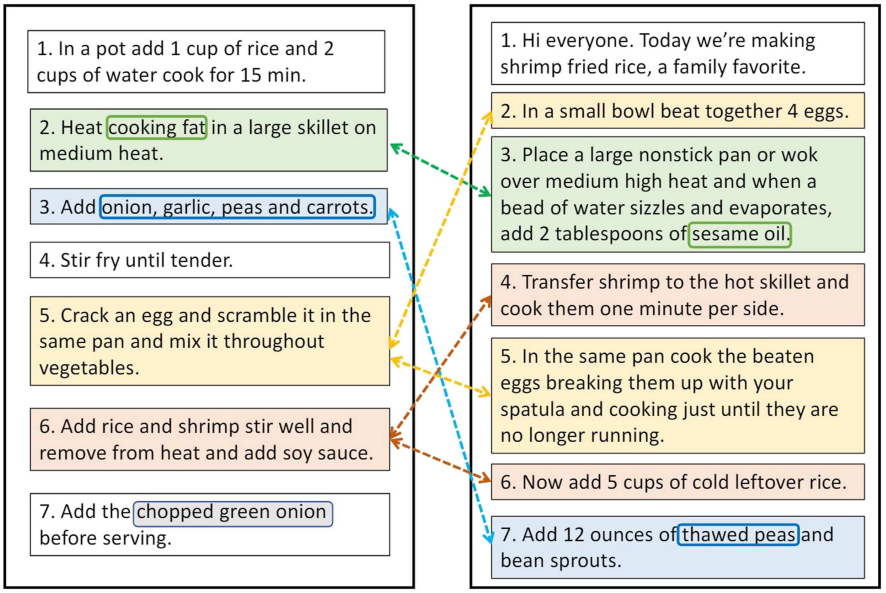


Figure 8
Recipe Alignment Example from Lin et al. (2020).

Most datasets lack hand-annotated alignments, driving reliance on unsupervised methods: HMMs (Lin et al. 2020), semi-supervised clustering (Wanzare et al. 2017), and neural encoders (BERT, LSTM) (Nandy, Kapadnis et al. 2023; Donatelli et al. 2021). Evaluation uses accuracy, precision, recall, and F1 against manually annotated subsets or held-out gold data (Regneri et al. 2010; Donatelli et al. 2021; Lin et al. 2020).

Limitations and Open Questions. Approaches to recipe action alignment achieve 66.97–72.4% accuracy (Donatelli et al. 2021; Nandy, Kapadnis et al. 2023), maintaining a 6.6–12% gap from human performance (79%; Donatelli et al. 2021). This gap concentrates in cases requiring commonsense inference (Donatelli et al. 2021): implicit actions (24% of disagreements) and light verb constructions (22% of disagreements). Analysis of these failures suggests two underlying problems: (1) local context bias may prevent capturing step-wise contexts across entire recipes (Nandy, Kapadnis et al. 2023); and (2) light verb constructions like "let rest" create inconsistent action boundaries, with 22% of human disagreements stemming from whether causative verbs constitute separate actions (Donatelli et al. 2021). These failures exemplify the implicit knowledge problem discussed in §5.3 and §6.1, creating an immediate opening to explore the potential of recent large language models for this task.

5.1.3 Detecting and Correcting Implicit Instructions. An intriguing but understudied challenge in procedural text is the presence of implicit or unclear statements. Anthonio, Bhat, and Roth (2020) address this by constructing a corpus based on revisions to WikiHow articles, focusing on whether sentences need clarification (e.g., adding missing pronouns or quantifiers). They also introduce a shared task (Roth and Anthonio 2021)

to promote research on implicit language, and initiate a workshop series.[7] Zeng et al. (2020) tackle the challenge of *repairing procedural text* by using external activity templates to fill in missing information. Zhang, Yamakata, and Tajima (2021) develop a dataset of Chinese recipes from the Haodou recipe Web site, where missing entities are supplemented using both text and images. Bisk et al. (2019) introduce the KIDSCOOK dataset, pairing cooking instructions with hierarchical sub-steps, and proposed a task to infer hierarchical relations between instructions and sub-steps. Similarly, Feng et al. (2022) create the R2R-ULN dataset, which involves guiding an agent through environments with underspecified navigation instructions. The works in this task fall into two main categories: (1) *detecting and correcting implicit or missing information*, and (2) *clarifying procedural instructions*. The first category focuses on identifying and completing missing or implicit details in procedural texts, using inputs that often include underspecified texts, sometimes with images or process models. The output is a refined text, a simulated representation, or a predicted completion, where missing actions, entities, or arguments are corrected, inferred, or predicted (Clark et al. 2018; Cheng and Erk 2018; Zeng et al. 2020; Zhang, Yamakata, and Tajima 2021). The second category aims to improve clarity and structure in instructional texts by refining vague or ambiguous instructions. The output is a clearer, explicit, and more actionable set of instructions or a prediction of where clarification is needed to achieve this (Anthonio, Bhat, and Roth 2020; Roth and Anthonio 2021; Bisk et al. 2019; Feng et al. 2022; Zhang, Wu, and Cai 2023).

Works on implicit information use pretrained language models and vision-language models (Roth and Anthonio 2021; Feng et al. 2022). These methods are typically evaluated using precision, recall, F1, and accuracy (Cheng and Erk 2018; Zhang, Yamakata, and Tajima 2021; Zeng et al. 2020).

Limitations and Open Questions. Performance splits sharply between relatively *explicit* clarification phenomena and more *implicit* ones: UnImplicit systems reach 92.7% on pronoun resolutions but hover near chance for modifier/quantifier insertions (53.6%–54.2% vs. 50% random) (Roth and Anthonio 2021). Similar patterns recur across procedural benchmarks where crucial content is *expected but not stated*: Models struggle when success requires recovering omitted or underspecified steps, arguments, or action effects rather than matching surface cues (Jiang et al. 2020; Zeng et al. 2020; Roth and Anthonio 2021). We identify two factors to be particularly important: (1) *Reference under state change:* Entities are repeatedly transformed or combined, so resolving "what does this refer to now?" becomes a state-tracking rather than standard coreference problem (Jiang et al. 2020). (2) *Sparse, indirect supervision for absence:* Supervision is dominated by observed mentions and edits, while events and arguments that *should* be present are only indirectly observable via revision filters or model–text misalignments, leading to skewed learning (Roth and Anthonio 2021; Anthonio, Bhat, and Roth 2020; Zeng et al. 2020; Jiang et al. 2020).

These challenges show up both in clarification-from-revision settings and in omission-recovery frameworks where only a subset of activities or entities is missing from the surface text. Moreover, revision-derived supervision risks conflating *data convenience* with *instructional adequacy*: Edits may reflect style or correctness rather than necessity, and UnImplicit's "unchanged $\Rightarrow$ no clarification" assumption is explicitly a

7 https://roth-anthonio-2021-unimplicit.github.io/.
https://roth-anthonio-2021-unimplicit2022.github.io/.

heuristic (Anthonio, Bhat, and Roth 2020; Roth and Anthonio 2021). A central open question is therefore how to define and evaluate "implicitness" relative to user expertise, task goals, and external context, not just observed edits or local textual cues (Roth and Anthonio 2021; Anthonio, Bhat, and Roth 2020; Jiang et al. 2020). This task most directly exposes the implicit knowledge problem central to procedural understanding, discussed in §5.3 and §6.1.

5.1.4 Entity State Tracking. In §4.3, we present a detailed analysis of how entities, procedures, and their relationships are represented in the literature. The entity (state) tracking task is defined on these representations and focuses on tracking entities (e.g., ingredients, tools) and their properties (e.g., color, location) and interactions through each step. Several datasets have been created, each focusing on different domains, e.g., hardware and software setup (Goyal et al. 2021), WikiHow (Tandon et al. 2020; Zhang et al. 2024), and cooking recipes (Rim et al. 2023). Entity tracking is mostly the primary task (Clark et al. 2018; Dalvi et al. 2018, 2019; Zhang et al. 2021b; Long, Pasupat, and Liang 2016; Li, Nye, and Andreas 2021; Kim and Schuster 2023; Bosselut et al. 2018; Zhang et al. 2024), or serves as an intermediate step for other event related tasks (Zhang et al. 2023b; Rim et al. 2023; Cheng and Erk 2018; Kazeminejad and Palmer 2023; Nandy, Kapadnis et al. 2023; Nandy, Kulkarni et al. 2024). For the first category, Dalvi et al. (2018) and Clark et al. (2018) leverage scientific procedural text, i.e., Process Paragraph (ProPara), while Zhang et al. (2021b) and Bosselut et al. (2018) use Recipes to model entity movements and state changes across multiple steps. These datasets use a predefined set of entities, properties, and actions, offering a more controlled setup. More recently, open-domain tracking datasets (Dalvi et al. 2019; Tandon et al. 2020) are introduced where entities and properties need to be *extracted or inferred*, which provides a more challenging setup. On the other hand, several others (Long, Pasupat, and Liang 2016; Kim and Schuster 2023) build simple synthetic datasets for a predefined set of actions and states. For the second category, entity tracking plays a supporting role for other tasks like causal reasoning, coreference resolution, and multi-task learning. Tracking how entities change state helps identify relationships between actions and their effects (Dalvi et al. 2019) or predict hypothetical event outcomes (Zhang et al. 2023b) for causal reasoning. Tracking also helps resolving coreferences (Rim et al. 2023) and identifying implicit arguments (Cheng and Erk 2018). Furthermore, tracking supports action alignment, step prediction, and understanding event-driven state changes (Kazeminejad and Palmer 2023; Nandy, Kapadnis et al. 2023; Nandy, Kulkarni et al. 2024).

Approaches range from tailored architectures like Neural Process Networks (Bosselut et al. 2018) to pretrained models (Zhang et al. 2021b; Nandy, Kapadnis et al. 2023), with more recent work integrating symbolic reasoning via ConceptNet and VerbNet (Kazeminejad and Palmer 2023). Evaluation uses accuracy, F1, and—for open-vocabulary settings—generation metrics like BLEU and BERTScore (Tandon et al. 2020; Zhang et al. 2024).

Limitations and Open Questions. Controlled experiments expose severe limitations in current models' entity tracking abilities. Probing classifiers achieve 86.8% accuracy on trivial cases but collapse to **3.1% on non-trivial tracking** (Kim and Schuster 2023). Even GPT-3.5 degrades from roughly 70–75% to about 25% accuracy as the number of state-changing operations *affecting a single entity* increases to seven, despite the entire description fitting within the context window (Kim and Schuster 2023). These failures are consistent with vanilla Transformers lacking persistent state mechanisms for updating entity representations over sequences of operations (Kim and Schuster 2023; Fagnou

et al. 2024). Notably, only code-pretrained models succeed at tracking: Kim and Schuster (2023) report that "only models in the GPT-3.5 series, which have been trained on both text and code, are able to perform non-trivial entity tracking," suggesting that text-only pretraining is insufficient for this capability to surface. Complementary work on open-vocabulary procedural state tracking finds that even strengthened baselines with explicit temporal dependency and entity awareness leave state-of-the-art models far from competent, especially when tracking many attributes over multiple steps (Wu, Li, and Ji 2023; Li and Huang 2023; Zhang et al. 2024).

These findings suggest a research direction: *Do transformers require explicit persistent-state mechanisms to track entity attributes compositionally?* The degradation with increasing state-changing operations—despite the full description fitting within the context window—indicates that access to tokens alone is insufficient for reliable state updating (Kim and Schuster 2023). This aligns with theoretical and empirical evidence that vanilla attention can be structurally inefficient for repeated entity updates, motivating modified attention/state-propagation mechanisms (Fagnou et al. 2024). More broadly, the continued difficulty of open-vocabulary procedural tracking—especially with many attributes over multiple steps—suggests evaluating memory-augmented and structured-state variants under the same stressors used by OpenPI-style benchmarks (Tandon et al. 2020; Zhang et al. 2024; Li and Huang 2023).

5.1.5 Instruction Parsing. Parsing natural language instructions into structured data representations such as trees, graphs, and BPMN is one of the most crucial steps to be able to run/execute them. There are two main approaches: (1) Supervised parsing, where the model uses labeled data to train on **Process Structure Extraction (PSE)**, and (2) Semi-supervised or unsupervised parsing, referred to as **Process Structure Induction (PSI)**, explained in detail below.

Process Structure Extraction. In PSE, unstructured procedural text is parsed into structured formats like trees, graphs, or BPMNs. PSE plays a key role in domains like **business process management** by extracting actions, actors, and their relationships within task sequences. Figure 9 shows an illustration of the PSE problem from Qian et al. (2020). This task is rich with diverse domains like *cooking recipes* (Diwan, Batra, and Bagler 2020; Pan et al. 2020; Wang et al. 2022; Qian et al. 2020; Kiddon et al. 2015; Maeta, Sasada, and Mori 2015; Zhang, Yamakata, and Tajima 2022; Bhatt et al. 2024; Miglani and Yorke-Smith 2020; Feng, Zhuo, and Kambhampati 2018; Brach, Košťál, and Ries 2025), *household and everyday tasks* (Zhang et al. 2021a; Feng, Zhuo, and Kambhampati 2018; Rula and D'Souza 2023; Kourani et al. 2024; Zhou et al. 2022; Zhang et al. 2024a; Miglani and Yorke-Smith 2020), *technical and scientific procedures* (Mysore et al. 2019; Rula and D'Souza 2023; Miglani and Yorke-Smith 2020), and *general instructions* (Ito, Suzuki, and Aizawa 2020). Diwan, Batra, and Bagler (2020), Feng, Zhuo, and Kambhampati (2018), and Wang et al. (2022) extract the structure (ingredients, cooking techniques, and utensils) of recipes from the RecipeDB dataset (Batra et al. 2020), `cookingtutorials.com`, and the Recipe1M dataset (Salvador et al. 2017), respectively. On the other hand, several other works generate new datasets. Mysore et al. (2019) generate 230 synthesis procedures (discrete process steps taken to synthesize the target material) with labeled graphs that express the semantics of the synthesis sentences (actions, materials involved, conditions, etc.). Zhang et al. (2021a) collect the MSComplexTasks dataset that contains complex tasks from Wunderlist[8] with their sub-task graph, while Zhang, Yamakata,

8 `https://en.wikipedia.org/wiki/Wunderlist`.

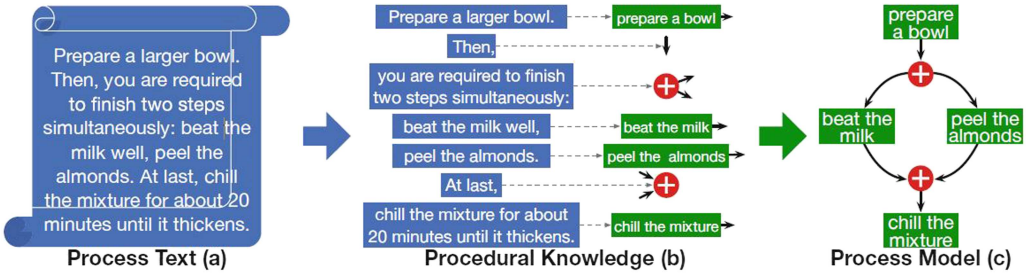


Figure 9
Illustration of the PME problem (Qian et al. 2020).

and Tajima (2022) create the MIAIS dataset (see §4). Graphs and trees are the most commonly used due to their relations with workflow representations (Zhang et al. 2021a; Pan et al. 2020; Mysore et al. 2019; Qian et al. 2020; Maeta, Sasada, and Mori 2015; Zhang, Yamakata, and Tajima 2022; Rula and D'Souza 2023; Zhou et al. 2022; Wang et al. 2022). Other works produce well-structured key-value pairs (Diwan, Batra, and Bagler 2020), PDDL (Miglani and Yorke-Smith 2020; Zhang et al. 2024a), or lists of structured sequential actions (Feng, Zhuo, and Kambhampati 2018).

Approaches evolved from basic NLP techniques (NER, POS tagging) (Diwan, Batra, and Bagler 2020) through neural architectures (Bi-LSTM, CNN) (Qian et al. 2020; Park and Motahari-Nezhad 2018) to Transformer models and LLMs for PDDL generation (Bhatt et al. 2024; Zhang et al. 2024a; Kourani et al. 2024). Evaluation uses precision, recall, F1, and, for executable outputs, PDDL solve rates (Zhang et al. 2024a).

Limitations and Open Questions. High local action-extraction scores do not guarantee reliable executable process models. Contextual extractors achieve 96.22% F1 on technical manuals but drop to 82.59% on home-and-garden instructions, indicating substantial performance variation across procedural domains (Miglani and Yorke-Smith 2020). Similarly, component-level correctness fails to ensure planning success: GPT-4o reaches only 21.4% intrinsic action accuracy and 45.3% problem solve rate, with preconditions (31.1–40.1%) notably harder to predict than effects (53.5–62.5%), consistent with the analysis that preconditions require deeper implicit knowledge about entity states and are often less explicitly stated in text (Zhang et al. 2024a). LLM-assisted BPMN generation frameworks can produce sound/executable models yet still show semantic deviations from intended process specifications, motivating iterative refinement via feedback (Qian et al. 2020). These extraction-to-execution gaps highlight that current systems capture surface procedural patterns while struggling with compositional generalization and implicit state reasoning, motivating the need for execution-grounded evaluation metrics discussed in §6.3.

Process Structure Induction. Induction uses unsupervised or semi-supervised techniques to *induce the latent* structure of event sequences. Here, the goal is to automatically identify and align similar event descriptions across multiple instances of a scenario/procedure, clustering them into paraphrase sets that represent the event types (nodes) within the procedure. Most common scripts are from everyday tasks, such as baking a cake, grocery shopping, or booking a flight, described as ESDs (Regneri et al. 2010; Wanzare et al. 2016, 2017), or with text narratives describing the process (Rula and

D'Souza 2023; Wu et al. 2023a; Liu et al. 2018). Additionally, some approaches incorporate structured data, such as graphs, to generalize subprocesses into more generic ones (Zhang et al. 2020). In addition to collecting processes from resources like WikiHow (Zhang et al. 2020; Wu et al. 2023a) and public manuals (Rula and D'Souza 2023; Wu et al. 2023a), other works (Wanzare et al. 2017; Regneri et al. 2010) use existing datasets, such as the OMICS corpus and the MiniWoB dataset (Shi et al. 2017). Similar to the process extraction task, graph-based structures dominate the field (Regneri et al. 2010; Wanzare et al. 2017; Wu et al. 2023a; Zhang et al. 2020; Wanzare et al. 2016). Several other works generate data in other formats, such as sequences of low-level actions (e.g., atomic or specific user actions, such as clicking buttons) (Liu et al. 2018) or ontologies (Rula and D'Souza 2023).

Traditional methods use Multiple Sequence Alignment (MSA) and clustering for temporal script graphs (Wanzare et al. 2016, 2017; Regneri et al. 2010), while recent work applies seq2seq models (Zhang et al. 2020) and GPT-4 for zero-shot structuring (Rula and D'Souza 2023). Evaluation relies on E-ROUGE, precision/recall, and crowdsourced annotation of subsets (Zhang et al. 2020; Wanzare et al. 2017).

Limitations and Open Questions. Models achieve only 14.8 E-ROUGE1 vs. 29.0 human performance at verb-level prediction, degrading to 3.5 vs. 11.6 when predicting complete event structures (all words) (Zhang et al. 2020). This degradation reveals a core bottleneck: Models identify individual steps but cannot assemble them into coherent structures. We believe two task-specific deficiencies might drive this: (1) *weak abstraction mechanisms*: Performance is highly sensitive to hyperparameter choices for abstraction depth (Zhang et al. 2020); and (2) *absence of explicit dependency modeling*: Current approaches lack representations of preconditions (Wu et al. 2023a) and step interchangeability (Wu et al. 2022). Current evaluation worsens the problem: Sparse reference orderings (only 1.7 ground truths per instance [Zhang et al. 2020]) fail to distinguish valid variations from genuine failures (Wanzare et al. 2016). The compositional reasoning limitations connect to architectural gaps discussed in §5.3 and §6.4.

5.1.6 Procedural Generation. Procedural generation, also referred to as script construction (Lyu, Zhang, and Callison-Burch 2021), similar to other tasks, like cooking recipes (Kiddon, Zettlemoyer, and Choi 2016; Liu et al. 2022; Lu et al. 2023; Nishimura et al. 2021), and everyday tasks (Modi et al. 2016; Sakaguchi et al. 2021; Lyu, Zhang, and Callison-Burch 2021; Nishimura et al. 2021; Sun et al. 2023; Nguyen et al. 2017; Yuan et al. 2023) like "baking a cake" or "fueling a car" from WikiHow and other sources, are the most common domains. While most datasets are derived from existing resources (Kiddon, Zettlemoyer, and Choi 2016; Nguyen et al. 2017; Rojowiec et al. 2020; Lyu, Zhang, and Callison-Burch 2021; Nishimura et al. 2021; Sun et al. 2023), several are created specifically for this task, such as proScript (Sakaguchi et al. 2021), CoScript (Yuan et al. 2023), WIKIPLAN and RECIPEPLAN (Lu et al. 2023), InScript (Modi et al. 2016), and XIACHUFANG (Liu et al. 2022) datasets. The output of the models vary, ranging from sequences of natural language steps (Rojowiec et al. 2020; Kiddon, Zettlemoyer, and Choi 2016; Lyu, Zhang, and Callison-Burch 2021; Sun et al. 2023; Liu et al. 2022; Yuan et al. 2023; Sakaguchi et al. 2021), to multimodal procedural plans pairing text and images (Lu et al. 2023; Nishimura et al. 2021), to graphing in DOT language—graph description language detailing event sequences (Sakaguchi et al. 2021). Input can also be diverse, such as descriptions of scenarios with ordered (Nguyen et al. 2017) or partially ordered (Sakaguchi et al. 2021) events, or just the procedure goal (Lyu, Zhang, and Callison-Burch 2021; Lu et al. 2023), with additional details like a list of ingredients

(Kiddon, Zettlemoyer, and Choi 2016), constraints (Yuan et al. 2023), or user preferences (Sun et al. 2023; Liu et al. 2022). Other input formats can include image sequences or pairs of "before" and "after" images (Rojowiec et al. 2020).

Approaches evolved from seq2seq models with LSTMs (Nguyen et al. 2017; Nishimura et al. 2021; Rojowiec et al. 2020) to pretrained LMs like T5 and GPT-3 (Sakaguchi et al. 2021; Yuan et al. 2023; Lyu, Zhang, and Callison-Burch 2021). Evaluation uses BLEU, ROUGE, BERTScore, and METEOR, though these metrics poorly capture procedural coherence (Lyu, Zhang, and Callison-Burch 2021; Yuan et al. 2023).

Limitations and Open Questions. Despite steady gains in fluency, current procedural generators still struggle for two recurring reasons. The first is *stateful consistency over long horizons*. Even when outputs are locally plausible, models often fail to maintain and update latent state as a procedure unfolds. This is visible in structured script generation on proScript, where graph edit distance remains far from human quality (model–human: 4.97 vs. 2.98; lower is better) (Sakaguchi et al. 2021). Similar issues appear at longer lengths. On LONGPROC, average performance degrades sharply with output length (GPT-4o: 83.4 at 2K vs. 38.1 at 8K) (Ye et al. 2025). A targeted ToM Tracking analysis at 8K reports that 19/20 inspected GPT-4o errors arise from incorrect long-range state inferences (Ye et al. 2025). SCEDIT similarly finds that edits often pass direct post-edit checks but fail to transfer to downstream script generation, with an average 27% drop when moving from post-edit success to script-level success (Li et al. 2025). The second recurring issue is *adaptation under interventions*. When procedures must change under substitutions or counterfactual conditions, models remain brittle. Counterfactual recipe generation highlights this brittleness under ingredient replacements, where models often fail to consistently incorporate replaced or added ingredients and to insert or delete ingredient-dependent actions (Liu et al. 2022). Together, these observations motivate approaches that make constraints and state explicit rather than leaving them implicit in next-token prediction. For example, constrained-generation pipelines such as over-generate-then-filter can substantially improve manual-evaluated script accuracy in controlled settings (e.g., 69%→95%) (Yuan et al. 2023), and suggest required paradigm shifts, discussed further in §6.2.

5.1.7 Question Answering. QA for procedural text typically requires an understanding of action sequences, causal relationships between steps, and implicit knowledge of the process described. Numerous datasets have been developed to support this task. For instance, datasets sourced from *WikiHow* include WikiHowQA (Bolotova-Baranova et al. 2023), the WikiHow dataset (Zhou, Shah, and Schockaert 2019), PARADISE (Uzunoglu, Şahin, and Safa 2024), and other WikiHow variants (Zhang, Lyu, and Callison-Burch 2020; Yang et al. 2021; Uzunoglu and Sahin 2023). *Everyday task* datasets include MCScript (Ostermann et al. 2018) and its successor, MCScript2.0 (Ostermann, Roth, and Pinkal 2019), along with HellaSwag (Zellers et al. 2019). Additional datasets address other domains, such as RecipeQA (Yagcioglu et al. 2018) for *recipes* and *social interactions* (Bauer and Bansal 2021). QA in procedural text either focuses on reasoning or reading comprehension. *Reading comprehension* benchmarks (Yagcioglu et al. 2018; Bolotova-Baranova et al. 2023; Zhou, Shah, and Schockaert 2019) mostly rely on retrieving correct information directly from the text without requiring deeper inferences. In contrast, *reasoning-based* benchmarks (Ostermann et al. 2018; Ostermann, Roth, and Pinkal 2019; Weston et al. 2016; Tandon, Dalvi et al. 2019; Zellers et al. 2019; Bauer and Bansal 2021; Yang et al. 2021; Wang et al. 2023) require models to go beyond merely extracting information, such as understanding causal relationships, making inferences (e.g.,

determining the effect of a missing step in a cooking recipe), predicting outcomes (e.g., what would happen if a particular process in a machine were altered), or applying commonsense knowledge to answer questions. Further categorization can be made based on whether the approach requires answering from a single document or multiple documents. **Single-document** retrieval involves tasks where models must derive answers from a single text or data source (Ostermann et al. 2018; Ostermann, Roth, and Pinkal 2019; Tandon, Dalvi et al. 2019; Zellers et al. 2019; Wang et al. 2023; Yagcioglu et al. 2018; Zhang, Lu, and Jaitly 2024; Zhou, Shah, and Schockaert 2019), while **multiple-document** retrieval (Bolotova-Baranova et al. 2023; Yang et al. 2021; Zhang, Lyu, and Callison-Burch 2020; Bauer and Bansal 2021) requires synthesizing information from various documents to answer questions.

Reading comprehension uses LSTM-based or fine-tuned transformer models (BERT, RoBERTa) (Zhou, Shah, and Schockaert 2019; Yagcioglu et al. 2018; Bolotova-Baranova et al. 2023), while reasoning tasks employ knowledge bases like ConceptNet (Bauer and Bansal 2021) and reinforcement learning for strategic reasoning (Zhang, Lu, and Jaitly 2024). Evaluation uses accuracy for multiple-choice and ROUGE/BLEU for abstractive QA (Ostermann et al. 2018; Bolotova-Baranova et al. 2023).

Limitations and Open Questions. Despite 70–94% accuracy on classification-style benchmarks (Zhang, Lyu, and Callison-Burch 2020; Tandon, Dalvi et al. 2019; Uzunoglu, Şahin, and Safa 2024), only 8–11% of predictions on TRIP are supported by fully correct reasoning chains (Storks et al. 2021). This is a stark gap: High task accuracy with near-zero verifiable reasoning reveals that models exploit surface correlations rather than understanding procedures. Three task-specific deficiencies emerge: (1) *keyword dependence*: dropping overlapping goal-candidate keywords induces 15–20 point accuracy drops (Uzunoglu, Şahin, and Safa 2024); (2) *unverified successes*: 31% of instances are correct but have entirely wrong underlying state chains (Storks et al. 2021); (3) *counterproductive supervision*: adding end-task supervision collapses verifiability from 9–11% to 0–1% (Storks et al. 2021). Furthermore, human–model gaps of up to 10–22 points persist on tasks requiring reasoning (essential-step detection, warning inference) (Wang et al. 2023; Uzunoglu, Şahin, and Safa 2024). These evaluation and reasoning gaps connect to §5.3 and §6.3.

5.1.8 Knowledge Acquisition/Mining. This task focuses on constructing comprehensive semantic knowledge structures by identifying and formalizing entities, actions, relationships, and processes from unstructured sources (e.g., natural language instructions). The goal is to enable AI systems to better reason, plan, and execute procedural tasks. Unlike process extraction and parsing, which focus on identifying sequential steps or syntactic structures, knowledge acquisition emphasizes building rich contextual understanding and relationships between concepts. Again WikiHow serves as the primary source (Chen et al. 2020; Jung et al. 2010; Chu et al. 2017; Steinert and Meneguzzi 2020; Sen et al. 2023; Miglani and Yorke-Smith 2020), in addition to scenarios and instruction manuals (Babli and Onaindía 2019; Miglani and Yorke-Smith 2020). Knowledge acquisition encompasses two main areas: (1) **Task Mining** extracts procedural knowledge (tasks, sub-tasks, methods) from instructional texts to support task-oriented conversational agents, recommendation systems, and knowledge bases (Sen et al. 2023; Chu et al. 2017; Chen et al. 2020; Jung et al. 2010), and (2) **Domain Acquisition** formalizes knowledge from natural language task descriptions into structured representations (e.g., PDDL models) for automated planning systems. These can be either static models

(Steinert and Meneguzzi 2020; Miglani and Yorke-Smith 2020) or dynamically adaptable ones that acquire new knowledge during execution (Babli and Onaindía 2019).

Earlier methods use rule-based systems and syntactic parsing (Chu et al. 2017; Steinert and Meneguzzi 2020; Babli and Onaindía 2019), while recent work uses pretrained models (BERT, LongFormer) with SRL (Chen et al. 2020; Sen et al. 2023). Task Mining evaluates against human annotations; Domain Acquisition tests PDDL executability in simulated environments (Steinert and Meneguzzi 2020; Jung et al. 2010).

Limitations and Open Questions. Performance remains severely limited: PDDL generation produces a valid plan for only 50% of instances (with just 0.97 solved actions on average per instance, i.e., actions whose parameters are available in the evaluated sentence) (Steinert and Meneguzzi 2020), while action identification achieves 21.21% recall@1 and object typing only 11.07% (Chen et al. 2020). These failures stem from fundamental data characteristics: 68.3% of action types and 88.2% of object types occur fewer than 10 times, creating a classic rare-data/long-tail regime (i.e., modeling long-tail distributions) where rare-but-valid actions are easily treated as noise (Chen et al. 2020). More critically, in around 91.2% of processes the action type labels differ from the predicates of associated events, and 84.2% of processes have object type labels that do not appear as event objects (Chen et al. 2020)—purely extractive approaches struggle to bridge this semantic gap. The task requires compositional generalization from descriptive language to formal representations, a capability current architectures lack. These limitations exemplify the implicit knowledge and compositional reasoning gaps discussed in §5.3 and §6.1.

5.2 Grounded Tasks

The final category of tasks we consider is **grounded tasks**, which aim to perform instructions that can be represented as a sequence of actions in a symbolic language on *simulated environments* (e.g., mobile phone, 3D simulation), external *knowledge resources* (e.g., querying a database, searching the Web), or in *structured or unstructured documents* through multi-turn task-oriented dialogues. These tasks typically require understanding and utilizing contextual information and domain-specific knowledge to effectively perform the tasks within specific domains, such as Web, mobile, game, robotics, and navigation. We build a taxonomy of environments that are commonly used in literature, namely, as Web (§5.2.1), Navigation/Household (§5.2.2), Robotic (§5.2.3), Game (§5.2.4), and GUI (§5.2.5); and provide the list of papers for each environment in Appendix C.

Meanwhile, *grounded task-oriented dialogue systems* build upon the principles of grounded tasks by enabling interactive, language-based guidance within various environments. These systems ground their responses in external sources such as documents (Feng et al. 2020, 2021; Strathearn and Gkatzia 2022) or real-time environmental data (Narayan-Chen, Jayannavar, and Hockenmaier 2019; Padmakumar et al. 2022). For instance, they can guide users to navigate in dynamic virtual environments (Narayan-Chen, Jayannavar, and Hockenmaier 2019; Padmakumar et al. 2022), or in troubleshooting and household management through static resources (Feng et al. 2020, 2021; Strathearn and Gkatzia 2022). Recent advancements in NLP, including LLMs and Retrieval-Augmented Generation (RAG) frameworks, have expanded the capabilities of these systems, making them essential in fields such as customer support and healthcare (Feng et al. 2021; Raghu et al. 2021). We discuss them in detail in §5.2.6. Furthermore, Table 5 provides a unified view of the diverse grounded task environments,

Table 5
Comparison of grounded task environments. SR = Success Rate, GCS = Goal-Condition Success, EM = Exact Match, LCS = Longest Common Subsequence.

Environment	State Space	Action Space	Observability	Key Metrics
Web	Web page content DOM past actions	Click type query navigate	Partial (POMDP)	SR EM Fuzzy Match
Navigation	Spatial layouts object locations	Camera move manipulate	Partial/Full	SR Path Length GCS LCS
Robotics	Robot pose objects joints	Navigate grasp manipulate	Partial	Task SR Sequence Completion
Game	Board/world state positions	Move pieces manipulate	Varies	Task Success Move Accuracy
GUI/Mobile	View hierarchy UI elements	Tap swipe drag	Partial	Task SR Accuracy
Dialogue	Utterances + doc/env state	Actions retrieval	Partial/Full	SR GCS EM F1 BLEU

comparing their structural properties (state/action spaces, observability) and evaluation metrics.

5.2.1 Web Environments. These environments mimic real-world Web interactions, tasking agents with activities from information seeking (Zhou et al. 2023; Deng et al. 2024) to e-commerce (Yao et al. 2022; Shi et al. 2017; Liu et al. 2018). It is generally defined as partially observable Markov decision processes (POMDP) (Zhu et al. 2024), where the state is the Web page content or past actions (Deng et al. 2023; Zhou et al. 2023); and actions are clicking, typing, and sending queries (Deng et al. 2023; Zhou et al. 2023; Liu et al. 2018).

Web agents perform step-by-step actions, interacting with DOM Trees, screenshots, or forms (Xu et al. 2021; Zhou et al. 2023; Shi et al. 2017), serving as benchmarks for language models (Zhou et al. 2023) and reinforcement learning (Shi et al. 2017; Liu et al. 2018). Methods range from BERT-based models (Xu et al. 2021) to GPT-4 (Deng et al. 2023) and WebGPT (Nakano et al. 2021), with RL approaches using binary success/failure rewards (Shi et al. 2017; Liu et al. 2018) or action-specific evaluation (Yao et al. 2022). Annotation granularity varies from command/navigation instructions (Mazumder and Riva 2021) to actions with HTML snippets (Deng et al. 2023). Evaluation uses exact/fuzzy match and success rates (Zhou et al. 2023; Deng et al. 2023; Shi et al. 2017).

Limitations and Open Questions. The 63-point gap between GPT-4 (14.41%) and humans (78.24%) on WebArena (Zhou et al. 2023) reveals fundamental architectural limitations rather than scaling issues. We suggest that three interacting factors might be driving this

failure. First, *context overflow*: Real-world HTML contains 7,000–14,000 tokens versus ∼500 in simulators, forcing lossy summarization that discards task-relevant information (Gur et al. 2024). Second, *error cascades*: Poor HTML understanding produces incorrect plans, which degrade state tracking in subsequent steps—a destructive feedback loop without closed-loop recovery mechanisms (Kim, Baldi, and McAleer 2023; Zhou et al. 2023). Third, *uncertainty blindness*: In the *UA-hint* setting—where the prompt explicitly tells the agent it may declare a task "unachievable"—GPT-4 erroneously identifies 54.9% of feasible tasks as impossible (an effect WebArena attributes primarily to the hint), causing premature termination rather than strategic exploration (Zhou et al. 2023). Finally, planning errors constitute the critical bottleneck: incorrect task decomposition causes most failures and compounds over multi-step trajectories (Gur et al. 2024). This suggests progress requires architectures with HTML-specific inductive biases and explicit uncertainty quantification: the challenges we return to in §6.5 and §6.6.

5.2.2 2D/3D Navigation/Household Environment. This environment involves simulated settings where autonomous agents interpret natural language instructions to execute navigation tasks (Shi et al. 2024). VirtualHome (Puig et al. 2018) is one such environment, characterized by spatial layouts agents must navigate, often using maps or virtual simulations (Shi et al. 2024). Common tasks are household assistance (Puig et al. 2018; Wu et al. 2024) and virtual navigation (Rajvanshi et al. 2023), with some environments focused solely on navigation (Qi et al. 2020; Li et al. 2022c; MacMahon, Stankiewicz, and Kuipers 2006; Zang et al. 2018; Artzi and Zettlemoyer 2013; Anderson et al. 2017) and others on complex, multi-step tasks in household settings (Shridhar et al. 2020a; Puig et al. 2018; Li et al. 2024; Zhou, Yin, and Neubig 2022; Feng et al. 2022; Misra et al. 2018; Das et al. 2017). Environments vary in structure and focus: REVE-CE (Li et al. 2022c) and REVERIE (Qi et al. 2020) emphasize visual navigation, while others use linguistic-driven navigation, guiding agents with natural language instructions (Vogel and Jurafsky 2010; Artzi and Zettlemoyer 2013; Zang et al. 2018). Meanwhile, VirtualHome (Puig et al. 2018; Wu et al. 2024) and FAVOR (Li et al. 2024) involve object manipulation and rearrangement through detailed instructions and AR-driven systems. Some, like ALFRED (Shridhar et al. 2020a), integrate both visual and linguistic inputs to enhance navigation and interaction capabilities (Misra et al. 2015; Zhou, Yin, and Neubig 2022). The studies differ in terms of *observability*, with some, like VirtualHome (Puig et al. 2018), offering *complete map awareness*, while others, like ALFRED (Shridhar et al. 2020a), present *partial observability* where agents must explore and adapt (Das et al. 2017; Li et al. 2022c; Qi et al. 2020; MacMahon, Stankiewicz, and Kuipers 2006; Anderson et al. 2017; Feng et al. 2022; Vogel and Jurafsky 2010). This dynamic adaptation makes tasks more challenging as agents gather new information and encounter obstacles (Li et al. 2022c; Qi et al. 2020; Shridhar et al. 2020a). The *action space is often limited* and repeated across environments, involving tasks like camera adjustments, moving through spaces, and manipulating objects (Ou et al. 2024; Anderson et al. 2017; Qi et al. 2020; Vogel and Jurafsky 2010; Li et al. 2022c; Puig et al. 2018; Li et al. 2024). Task complexity in environments like REVERIE (Qi et al. 2020) and REVE-CE (Li et al. 2022c) is increased by adding object identification or manipulation (Li et al. 2024; Puig et al. 2018; Shridhar et al. 2020a).

Methods have evolved from sequence-to-sequence models with attention and reinforcement learning (Anderson et al. 2017; Qi et al. 2020) to LLM-based planners combining language models with hierarchical planning (Song et al. 2023; Rajvanshi et al. 2023); see Lin et al. (2023) for detailed coverage. Evaluation uses success rate and path length (Anderson et al. 2017; Shridhar et al. 2020a), with variations including

navigation-plus-object-identification in REVERIE (Qi et al. 2020), Longest Common Subsequence in VirtualHome, and path-weighted success rates in ALFRED (Shridhar et al. 2020a).

Limitations and Open Questions. A large gap remains between human performance on ALFRED (91% task success on test-unseen) and automated agents (Shridhar et al. 2020a). The original ALFRED baselines achieved only 2.1–4.0% task success on test-seen and about 0.4–0.5% on test-unseen (Shridhar et al. 2020a). More recent *LLM-based* agents substantially improve performance (e.g., OPEx-S reports 44.55% on test-seen and 42.25% on test-unseen) (Shi et al. 2024), yet still fall short of human success. Moreover, OPEx shows that oracle perception (perfect depth and instance segmentation/object masks) yields a large boost (e.g., 59.43% success on valid-unseen), but still does not close the gap (Shi et al. 2024). Sub-goal analysis reveals a striking asymmetry: Agents achieve approximately 90% success on object-agnostic sub-goals like heating or cooling items (Shridhar et al. 2020a), but performance is much lower for navigation-dependent sub-goals such as going to and picking up objects (about 51% and 32% on seen, dropping to about 22% and 21% on unseen) (Shridhar et al. 2020a). This pattern suggests that agents learn basic interactions but lack the *spatial reasoning* and *long-horizon planning* needed to reliably localize targets, maintain spatial context over time, and recover from partial observability. These navigation-heavy failures are consistent with limitations in building and using structured spatial representations, and they relate to broader state-tracking and memory challenges discussed in §5.1.4.

5.2.3 Robotic Environments. We differentiate robotic environments from 2D/3D environments, as the former focuses on enabling real robots to understand and carry out tasks based on natural language instructions in dynamic settings (Howard, Tellex, and Roy 2014; Matuszek et al. 2012), whereas the latter refers to simulated environments. All surveyed papers in this category involve robots or robotic systems executable on robots (Nair et al. 2021). Key features include integrating sensory inputs like visual, tactile (Mees et al. 2021), and proprioceptive data (Nair et al. 2021), allowing robots (e.g., Franka Emika Panda robot arm [Mees et al. 2021; Singh et al. 2022; Nair et al. 2021], PR2 [Misra et al. 2014; Bucker et al. 2022], Baxter [Scalise et al. 2018], robotic lift [Tellex et al. 2011a; Matuszek et al. 2012], etc.) to interact effectively with their surroundings. Robots mostly operate in *partially observable* environments, interpreting complex language commands for tasks like *navigation, manipulation, and trajectory reshaping* (Bucker et al. 2022; Singh et al. 2022; Tellex et al. 2011a). State spaces typically include the robot's position, object locations, and environmental features. For example, Tellex et al. (2011a,b) and Misra et al. (2014) define locations and paths using semantic maps, while Mees et al. (2021) and others focus on object positions and robot joints (Bisk, Yuret, and Marcu 2016; Bucker et al. 2022). CALVIN (Mees et al. 2021) uses multiple cameras to capture sensory inputs, focusing on RGB data, while other papers incorporate depth images for multi-step tasks (Singh et al. 2022).

Methods combine probabilistic and neuro-symbolic approaches (Tellex et al. 2011b; Singh et al. 2022) using tools like Generalized Grounding Graphs (Tellex et al. 2011b), imitation learning and offline reinforcement learning (Nair et al. 2021; Mees et al. 2021; Cleveston et al. 2025), and more recently LLM-based planning with RL execution (e.g., SayCan [Ahn et al. 2022]). We refer the readers to Khan et al. (2020) and Tang et al. (2024) for comprehensive RL-robotics coverage. Evaluation focuses on task success rates, sequence completion, and trajectory prediction accuracy.

Limitations and Open Questions. Results across robotic manipulation benchmarks reveal three recurring limitations. (1) *Long-horizon execution collapses under multi-step instruction sequences.* In CALVIN, a representative imitation-learning baseline succeeds on 48.9% when executing a single instruction but drops to 0.08% when required to execute five instructions sequentially (Mees et al. 2021). In SayCan, overall execution success is 74% in the mock-kitchen setting, yet the long-horizon family achieves only 47% execution success (Ahn et al. 2022). These outcomes indicate that even when individual skills can be executed, chaining multiple language-specified subtasks remains unreliable and likely amplifies small errors over time. (2) *Robustness to environment variation remains limited.* In CALVIN, success decreases from 53.9% when training and testing in the same environment to 35.6% when training across multiple environments and testing in a held-out one (Mees et al. 2021). In SayCan, overall execution drops from 74% in a mock kitchen to 60% in a real kitchen (Ahn et al. 2022). Together, these results suggest that transferring language-conditioned behavior across setups is still sensitive to perception and low-level control details. (3) *Key linguistic phenomena and reference ceilings are under-specified.* SayCan explicitly notes that the base system does not handle negation (Ahn et al. 2022), and neuro-symbolic program learning reports drops in performance when relational concepts are present (Singh et al. 2022).

5.2.4 Game Environments. Simulating real-world challenges in game environments, e.g., strategic planning in chess (Toshniwal et al. 2021), collaborative building in Minecraft (Narayan-Chen, Jayannavar, and Hockenmaier 2019; Jayannavar, Narayan-Chen, and Hockenmaier 2020), and procedural level generation in educational games (Hooshyar et al. 2018), is common. We further divide game environments w.r.t. the task they focus on: **Structured, Strategic, and Predictive Environments** feature deterministic environments like chess and Othello (Toshniwal et al. 2021; Li et al. 2022a), where state, action, and observation spaces are defined by the positions and legal moves on the board. **Collaborative and Interactive Virtual Worlds** like Minecraft involve 3D environments where agents manipulate blocks and navigate using a limited view of the world (Narayan-Chen, Jayannavar, and Hockenmaier 2019; Jayannavar, Narayan-Chen, and Hockenmaier 2020). These environments focus on collaborative tasks. **Abstract and Procedural Knowledge Environments** such as HEXAGONS (Lachmy et al. 2022) and ScriptWorld (Joshi et al. 2023) emphasize abstract reasoning and procedural knowledge. In ScriptWorld, tasks are defined by sequential textual scenarios, while HEXAGONS focuses on tile painting within a hexagonal grid. **Adaptive and Text-based Environments** like TextWorld (Marc-Alex et al. 2018) and SmartPlay (Wu et al. 2023b) challenge AI systems with tasks requiring spatial reasoning and decision-making. TextWorld involves text-based navigation, while SmartPlay evaluates agents across diverse mini-games, including Rock-Paper-Scissors and Minecraft-like environments. Methods vary by environment: Chess (Toshniwal et al. 2021) and Othello (Li et al. 2022a) use the GPT-2 architecture and Othello-GPT, respectively, for move prediction; Minecraft uses Seq2Seq-style models for the Builder Action Prediction (BAP) task (Narayan-Chen, Jayannavar, and Hockenmaier 2019); HEXAGONS (Lachmy et al. 2022) uses DeBERTa and T5; TextWorld (Marc-Alex et al. 2018) tests RL agents like LSTM-DQN; and SmartPlay (Wu et al. 2023b) evaluates LLM families (e.g., GPT-style models and open-weight families such as LLaMA and Mistral), including instruction-tuned variants, across games. Evaluation includes task success and communication efficiency (Narayan-Chen, Jayannavar, and Hockenmaier 2019).

Limitations and Open Questions. Performance benchmarks reveal fundamental gaps in interactive reasoning and/or world state tracking: GPT-4 achieves only 26% of human performance on complex planning (Crafter), and 61% on 3D navigation (Minecraft). While transformers reach 97.7% legal move accuracy in chess, exact move prediction drops to 52% (Wu et al. 2023b; Toshniwal et al. 2021). In procedural instruction execution, most steps (>60%) contain abstract instructions, with 50% at mid-to-high abstraction levels that models struggle to execute correctly (Lachmy et al. 2022). Similar abstraction demands also arise in script-based environments (Joshi et al. 2023). We hypothesize that three key problems underlie these failures. *Spatial reasoning*: Models generate contradictory directional commands and lose fine-grained spatial grounding in text-based descriptions (Wu et al. 2023b; Narayan-Chen, Jayannavar, and Hockenmaier 2019). *Inadequate world models*: Linear probes exhibit $\geq$20% error versus single-digit error for nonlinear probes (down to $\sim$1.7%), suggesting that representational structure may be present but not linearly accessible (Li et al. 2022a). The gap between 97.7% legal move generation and 52% exact move accuracy suggests that satisfying local rule constraints can coexist with difficulty in selecting strategically appropriate actions, implicating limits in interactive reasoning and/or world-state tracking[9] (Toshniwal et al. 2021), while HEXAGONS error rates increase substantially from concrete to high-abstraction instructions (Lachmy et al. 2022). *Attention limitations*: Even in domains like chess, restricting attention to 50-token windows degrades legal move accuracy by $\sim$2 points (97.7%$\rightarrow$95.8%), with approximate attention (Reformer) dropping to 88.0% (Toshniwal et al. 2021).

These results expose a gap between surface-level pattern learning and robust world modeling—while models may encode latent state, they often fail to reliably use it for compositional spatial and strategic reasoning.

5.2.5 GUI Environments. Mobile environments for grounded task execution, also known as GUI environments, enable AI agents to interact with mobile app interfaces through actions like tapping, swiping, and navigating to perform a diverse set of tasks. Similar to Web environments, mobile environments focus on GUI interactions but extend to a broader range of applications, from system-level tasks to diverse fields like education (Burns et al. 2022), business (Banerjee et al. 2023; Zhang et al. 2023a), and entertainment (Li et al. 2020). Key datasets and platforms for mobile environments share a focus on vision-language navigation and UI interaction. MoTIF (Burns et al. 2022) provides 6,100 natural language tasks with action localization, while AndroidEnv (Toyama et al. 2021) offers a reinforcement learning platform for Android, with task-specific annotations. UICaption (Banerjee et al. 2023) pairs UI images with captions, though it lacks event annotations. PIXELHELP, ANDROIDHOWTO, and RICOSCA (Li et al. 2020) map natural language to UI actions. Mobile-Env's WikiHow Task Set (Zhang et al. 2023a) offers a benchmark for LLM-based agents with detailed annotations for cross-page navigation and QA tasks.

Action, state, and observation spaces in mobile environments reflect real app interactions. MoTIF (Burns et al. 2022) and Mobile-Env (Zhang et al. 2023a) define the state space by app view hierarchy, and actions like tapping and swiping. On the other hand, AndroidEnv (Toyama et al. 2021) focuses on pixel-based reinforcement learning

9 We use **world-state tracking** to mean maintaining a consistent latent representation of the underlying environment state (e.g., a chess board) as it evolves through actions, and **interactive reasoning** to mean selecting actions by anticipating multi-step consequences under that evolving state.

interactions, while Lexi (Banerjee et al. 2023) expands UI understanding to both mobile and desktop. Observation spaces also vary, including UI object hierarchies (Li et al. 2020), pixels, and screenshots (Banerjee et al. 2023), while actions involve diverse UI manipulations like dragging and typing. Finally, platforms like AndroidEnv (Toyama et al. 2021) and Mobile-Env (Zhang et al. 2023a) simulate real-world operations on actual devices, including Pixel phones for realistic evaluations (Li et al. 2020), while MoTIF (Burns et al. 2022) focuses on task feasibility but does not extend to real-world testing on physical devices, concentrating on controlled simulations.

Models include GPT-3.5-turbo, GPT-4, LLaMA 2 (Zhang et al. 2023a), and vision models like ViLBERT (Banerjee et al. 2023). Evaluation uses accuracy, recall, completion rates, and rewards (Zhang et al. 2023a; Toyama et al. 2021).

Limitations and Open Questions. Even the strongest agents (GPT-4, AgentLM-70B) achieve only 30–43% success on Mobile-Env's WikiHow tasks, with GPT-4 dropping to ≈7.5% on QA (Zhang et al. 2024). These failures stem from compounding weaknesses: *Feasibility blindness*: MoTIF's classifier misclassifies 44% of infeasible tasks as feasible, causing agents to attempt impossible goals (Burns et al. 2022). *Representation mismatch*: Generic CLIP encoders on screen images outperform domain-specific Screen2Vec on view hierarchies (58.2 vs. 33.7 F1), suggesting existing GUI representations may fail to capture functional semantics (Burns et al. 2022). *Visual grounding brittleness*: GPT-4V achieves only 3–10% success on pixel-level action prediction, improving only with element-level detection aids (Zhang et al. 2024). A key next step is to build agents that can recognize infeasibility early and recover gracefully, and to develop UI representations that support compositional, cross-app functional generalization rather than relying on pixel- or layout-level cues. We find the GUI domain to be intrinsically challenging: Identical icons serve different functions across apps, and dissimilar widgets implement the same operation, undermining assumptions in vision-language models trained on natural images (Banerjee et al. 2023; Burns et al. 2022). These patterns reveal that LLM planning advances do not transfer directly to GUI settings, where representation failures and brittle grounding dominate. Progress likely depends on training signals and interaction scaffolds that make grounding robust to UI variability, as well as benchmarks that disentangle planning mistakes from perception/grounding failures so improvements can be targeted.

5.2.6 Grounded Task-Oriented Dialogue. Task-oriented dialogue (TOD) systems facilitate specific tasks through conversational interactions. These systems use underlying resources such as documents, databases, or live environmental data to ensure accurate and contextually relevant responses. Researchers have extended TOD systems across various domains, tailoring them to function effectively within interactive and embodied environments, such as guiding users in navigation, construction, or household chores within simulated or virtual settings (Gao et al. 2022; Srinet et al. 2020; Narayan-Chen, Jayannavar, and Hockenmaier 2019; Padmakumar et al. 2022; Moghe et al. 2024). In customer support, TOD systems enhance technical assistance or customer service by utilizing documents or flowcharts (Feng et al. 2021; Strathearn and Gkatzia 2022; Raghu et al. 2021; Feng et al. 2020; Chen et al. 2021). Additionally, specialized TOD systems cater to specific needs, such as managing cooking recipes (Jiang et al. 2022), supporting data visualization (Shao and Nakashole 2020), simulating interactions in mobile GUIs (Sun et al. 2022), and facilitating storytelling within virtual worlds (Ammanabrolu and Riedl 2021).

Data collection for these dialogue systems is generally expensive, as it typically requires two users to engage in a conversation. Researchers have adopted several approaches to address this challenge. For instance, Wizard-of-Oz methods have been used in datasets such as META-GUI (Sun et al. 2022), ChartDialogs (Shao and Nakashole 2020), and CookDial (Jiang et al. 2022). Human–human interaction is used in the creation of datasets like Minicraft (Narayan-Chen, Jayannavar, and Hockenmaier 2019), ABCD (Chen et al. 2021), Task2Dial (Strathearn and Gkatzia 2022), and TEACh (Padmakumar et al. 2022). Additionally, human annotation is used in datasets such as CraftAssist Instruction Parsing (Srinet et al. 2020), DialFRED (Gao et al. 2022), Doc2Dial (Feng et al. 2020), MultiDoc2Dial (Feng et al. 2021), and FLONET (Raghu et al. 2021).

Systems can be grounded on *documents* and other static data sources, such as flowcharts and knowledge bases (Moghe et al. 2024; Shao and Nakashole 2020; Jiang et al. 2022; Chen et al. 2021; Feng et al. 2021; Strathearn and Gkatzia 2022; Feng et al. 2020; Raghu et al. 2021; Ammanabrolu and Riedl 2021), where the model has user utterances and grounding documents as input, to generate a sequence of actions or commands. Alternatively, systems can be grounded on the *environment* (Gao et al. 2022; Srinet et al. 2020; Sun et al. 2022; Narayan-Chen, Jayannavar, and Hockenmaier 2019; Padmakumar et al. 2022), where the input consists of user utterances plus real-time environmental state, and the output is dynamically responsive sequences of actions performed within that environment. This allows for interactions based on navigating physical spaces or manipulating objects within simulations.

Transformer-based models dominate, with BERT for retrieval and action tracking (Feng et al. 2020, 2021; Jiang et al. 2022; Chen et al. 2021; Sun et al. 2022; Padmakumar et al. 2022), Seq2Seq for instruction generation and parsing (Narayan-Chen, Jayannavar, and Hockenmaier 2019; Gao et al. 2022; Srinet et al. 2020; Shao and Nakashole 2020; Ammanabrolu and Riedl 2021), and RAG for document-grounded response generation (Feng et al. 2021; Raghu et al. 2021). Evaluation uses Exact Match and F1 for span selection (Feng et al. 2020, 2021), BLEU for response quality (Shao and Nakashole 2020; Feng et al. 2020; Chen et al. 2021), and Success Rate/Goal Completion for task achievement (Gao et al. 2022; Padmakumar et al. 2022; Raghu et al. 2021; Chen et al. 2021). See Appendix B for dataset details.

Limitations and Open Questions. Benchmarks expose systematic, not marginal gaps: ABCD shows 50.8 points between RoBERTa-Large (31.9%) and humans (82.7%) (Chen et al. 2021); TEACh collects sessions with a 74.17% human success rate, while baseline models achieve 4.8–7.06% success on the EDH benchmark (Padmakumar et al. 2022). These gaps likely reflect three intertwined failure modes. *Structure blindness*: The Episodic Transformer drops from 38.24% on single-instruction ALFRED to 5–7% on TEACh's multi-turn dialogues; TEACh further uses unimodal ablations to test whether models are simply memorizing action sequences, suggesting flat sequence models struggle to learn hierarchical task structure (Padmakumar et al. 2022). *Retrieval brittleness*: FLODial achieves only 66.1% Recall@1 on unseen flowcharts, with 45.5% of errors on non-neighboring nodes, showing retrieval-augmented systems fail to reason over graph topology (Raghu et al. 2021). *Difficulty with abstraction*: Rule-based agents achieve 0% success on several compositional TEACh tasks despite 150 hours of engineering; DialFRED shows generalization drops from 25.4–47.8% (seen) to 18.3–33.6% (unseen) (Padmakumar et al. 2022; Gao et al. 2022).

The compositional nature of grounded dialogue—requiring simultaneous procedural grounding, long-horizon planning, multi-modal integration, and commonsense reasoning (e.g., TEACh averages $\sim$165 actions per session with $\sim$14 utterances, with

irreversible operations like slicing and toasting) (Padmakumar et al. 2022; Strathearn and Gkatzia 2022)—suggests that isolated component improvements are insufficient. Both supervised models and hand-crafted rules lack mechanisms for emergent hierarchical abstraction. They also lack robust error detection and recovery strategies that humans utilize naturally. Together, these results point to the need for integrated end-to-end solutions rather than isolated component improvements. This raises a central open question: in end-to-end settings, which failures are primary versus downstream effects, and what mechanisms enable hierarchical abstraction and reliable recovery under irreversible state changes?

5.3 Cross-Cutting Challenges

While the preceding sections document task-specific findings, several fundamental limitations recur across benchmarks. Table 6 summarizes these six challenges; the paragraphs below synthesize evidence from multiple tasks and offer our interpretation of why these limitations persist, necessarily bounded by the solutions attempted thus far.

5.3.1 The Implicit Knowledge Problem. Consider a repair manual that instructs users to "reassemble in reverse order." A human reader effortlessly expands this into the specific sequence of steps performed earlier, now inverted. Or consider a recipe that says "sauté until fragrant": Readers infer appropriate heat levels, timing, and what "fragrant" means for onions versus garlic. Procedural text is saturated with such implicit content: Studies of entity tracking find that roughly 40% of *referred-to* entities are unmentioned in the text (Tandon et al. 2020); analyses of knowledge acquisition show that a large majority of action labels require inference beyond surface lexical matching (Chen et al. 2020); and event alignment work reveals systematic failures on functionally equivalent actions

Table 6
Cross-cutting challenges in procedural text understanding. Each challenge recurs across multiple tasks, suggesting fundamental limitations rather than task-specific difficulties.

Challenge	Core Problem	Affected Tasks
Implicit Knowledge	Cannot reason about information that is expected but not explicitly stated	Entity Tracking (Tandon et al. 2020), Knowledge Acquisition (Chen et al. 2020; Steinert and Meneguzzi 2020), Alignment (Donatelli et al. 2021; Wanzare et al. 2017), Implicit Detection (Roth and Anthonio 2021; Anthonio, Bhat, and Roth 2020)
State Maintenance	Performance degrades sharply as the number of steps increases	Entity Tracking (Kim and Schuster 2023; Tandon et al. 2020), Robotics (Mees et al. 2021), Navigation (Shridhar et al. 2020a; Shi et al. 2024)
Evaluation	High benchmark accuracy does not reflect genuine procedural understanding	QA (Storks et al. 2021; Uzunoglu, Şahin, and Safa 2024), Process Extraction (Zhang et al. 2024a; Qian et al. 2020), Summarization (Kwon and Lee 2025; DeChant and Bauer 2022)
Compositional Generalization	Success on atomic operations does not transfer to composed sequences	Generation (Ye et al. 2025; Sakaguchi et al. 2021), Induction (Zhang et al. 2020), Dialogue (Padmakumar et al. 2022; Gao et al. 2022)
Grounding	Language understanding does not transfer to action execution	Web (Zhou et al. 2023; Gur et al. 2024), Navigation (Shridhar et al. 2020a; Shi et al. 2024), Robotics (Ahn et al. 2022; Mees et al. 2021), GUI (Burns et al. 2022; Zhang et al. 2024)
Error Accumulation	Lack mechanisms to detect and recover from intermediate failures	Web (Zhou et al. 2023; Kim, Baldi, and McAleer 2023), Robotics (Mees et al. 2021; Ahn et al. 2022), Dialogue (Padmakumar et al. 2022; Chen et al. 2021)

that differ only lexically (Donatelli et al. 2021). Tests that isolate implicit reasoning show a stark split: in the UnImplicit shared task, approaches achieve over 90% accuracy on pronoun-based clarifications but only up to 56% accuracy on harder categories such as modifiers, quantifiers, and modal verbs (Roth and Anthonio 2021).

Why does this remain difficult? We hypothesize that the core issue is representational: approaches (from sequence-to-sequence models through encoder-based transformers to large language models) learn to process tokens that are *present*, but procedural reasoning requires attending to what is *absent*. Training objectives that optimize for next-token prediction on observed text do not directly reward reasoning about expected-but-missing elements. Whether future approaches can address this through explicit world models, abductive reasoning modules, or training objectives that reward inference over implicit content remains an open question.

5.3.2 The State Maintenance Problem. Consider tracking a piece of dough through a bread recipe: It begins as "flour and water," becomes "the dough" after mixing, transforms into "the risen dough" after proofing, and finally "the loaf" after baking. Each reference points to the same physical entity in different states. This kind of state tracking is fundamental to procedural understanding, yet proves remarkably fragile across tested approaches. In controlled entity tracking experiments, accuracy on single-state cases exceeds 85%, but collapses below 5% when tracking entities through multiple state changes, even when the entire procedure fits within context limits (Kim and Schuster 2023). Robotic manipulation shows parallel degradation: 54% success on single instructions drops to below 1% on five-step chains (Mees et al. 2021).

The sharp shape of this degradation suggests that tested approaches perform step-by-step pattern matching without maintaining coherent state representations. One notable finding offers a clue: Only approaches pretrained on source code, where variables are explicitly declared and state updates syntactically marked, show improved entity tracking (Kim and Schuster 2023). This suggests that natural language procedural text may lack the structural cues needed to learn robust state maintenance from text alone, pointing toward training regimes that incorporate explicit state-update supervision or draw from programming language semantics.

5.3.3 The Evaluation Gap. A question-answering system might correctly predict that "the cake will be dry" if the eggs are omitted, but did it reason through the causal chain (eggs provide moisture and binding); or merely associate "omit ingredient" with "bad outcome"? Across procedural benchmarks, we find a troubling pattern: High task accuracy masks shallow reasoning. QA systems achieve 70–90% accuracy while producing verifiably correct reasoning chains less than 11% of the time (Storks et al. 2021). For process extraction, Proc2PDDL shows that intrinsic action prediction accuracy remains low (18–21%), and that executing predicted PDDL solves only 36–45% of planning problems (Zhang et al. 2024a). Summarization metrics reward lexical overlap but cannot detect step-ordering violations that destroy procedural coherence (Kwon and Lee 2025).

These patterns reveal that standard metrics reward surface similarity rather than procedural validity. More troubling, training directly on end-task objectives can *worsen* interpretable reasoning: One study found that adding task-specific supervision collapsed verifiable reasoning from roughly 10% to near zero (Storks et al. 2021). This suggests that approaches learn shortcuts that inflate benchmark scores while bypassing the procedural understanding those benchmarks were designed to measure.

5.3.4 The Compositional Generalization Gap. Consider the difference between knowing how to chop vegetables and how to make a stir-fry. The latter requires not just executing "chop," "heat oil," and "add ingredients" individually, but sequencing them correctly, tracking what has been added, and adjusting timing based on ingredient properties. Across procedural benchmarks, approaches that succeed on atomic operations fail when those operations must be composed. Process induction achieves reasonable performance on predicting individual verbs but degrades sharply when predicting complete event structures that combine verbs with arguments and temporal relations (Zhang et al. 2020). Dialogue systems collapse from 38% success on single instructions to 5–7% on multi-turn interactions (Padmakumar et al. 2022). Procedural generation degrades as output length increases, with detailed error analysis revealing that failures stem from incorrect state inferences across steps, rather than from forgetting earlier content (Ye et al. 2025).

Is this simply a memory or length limitation? Several findings suggest otherwise. The degradation in process induction occurs when moving from verbs to complete structures at the same length (Zhang et al. 2020). Generation failures are state-inference errors, not content-forgetting errors (Ye et al. 2025). The pattern suggests that tested approaches lack hierarchical abstraction: They cannot decompose complex procedures into subgoals, execute them while tracking intermediate states, and verify results before proceeding.

5.3.5 The Grounding Gap. An agent that can fluently describe how to navigate a Web site may nonetheless fail catastrophically when asked to actually book a flight. This gap between language understanding and action execution, which we term the grounding gap, persists across interactive environments. On Web navigation, household robotics, and embodied dialogue, the gap between automated approaches and human performance spans 60–85 percentage points (Zhou et al. 2023; Shridhar et al. 2020a; Padmakumar et al. 2022). What causes this gap? One hypothesis is perceptual: Perhaps language understanding is adequate but environmental perception fails. However, experiments providing oracle perception, including perfect object segmentation and ground-truth scene descriptions, show the gap remains largely unchanged (Shi et al. 2024). Analysis of navigation failures reveals a telling asymmetry: Agents achieve roughly 90% success on perception-light sub-goals (heating an object, cooling an object) but drop to 20–30% on navigation-dependent ones (finding objects, picking them up) (Shridhar et al. 2020a). Error analyses in robotic settings find that roughly 65% of failures stem from high-level planning mistakes, such as selecting incorrect actions or action sequences, rather than low-level execution errors (Ahn et al. 2022).

These findings suggest that approaches trained on text have learned associations between words and situations, but not the causal structure of how actions transform world states. They can describe what actions mean without modeling their effects, preconditions, or interactions.

5.3.6 The Error Accumulation Problem. A Web agent attempting to book a flight may click the wrong dropdown menu, causing subsequent actions such as selecting dates or choosing seats to operate on the wrong form entirely. Each step then compounds the initial error, and because the agent lacks a mechanism to detect the mistake, the failure propagates unchecked. This error accumulation is particularly damaging in procedural settings, where later steps depend on earlier ones.

The pattern appears across grounded environments. In Web navigation, incorrect actions change page state unpredictably, degrading subsequent action selection in a destructive feedback loop (Zhou et al. 2023; Kim, Baldi, and McAleer 2023). In robotic manipulation, failed grasps or incorrect placements cascade into unrecoverable trajectories (Mees et al. 2021). Perhaps most notably, when given the option to declare tasks unachievable, approaches show poor calibration: Over 50% false-negative rates lead them to abandon feasible tasks while persisting on impossible ones (Zhou et al. 2023). They cannot distinguish "this task cannot be done" from "I do not know how to proceed."

This contrasts sharply with human procedure-following, in which continuous progress monitoring, error detection, and plan revision are routine. Addressing error accumulation, therefore, likely requires explicit mechanisms for uncertainty estimation, state verification, and recovery, moving from open-loop action sequences toward closed-loop systems that detect and correct failures during execution.

6. Pointers to Future Research

The challenges identified in §5.3 point toward specific research directions. We frame each as a shift from current practice toward capabilities that would address fundamental limitations.

6.1 Recovering Implicit Knowledge

Learning to infer unstated information instead of **matching surface patterns**.
Addresses: Implicit Knowledge (§5.3.1) → Entity Tracking, Knowledge Acquisition, Alignment, Implicit Detection

Procedural text is rarely self-contained. Studies suggest that a substantial portion of relevant entities—up to 40% in some analyses—may never be explicitly mentioned (Tandon et al. 2020), and most action labels cannot be recovered through simple word matching alone (Chen et al. 2020). Instructions routinely assume readers know default values ("season to taste"), typical preconditions (heating a pan before searing), and domain conventions that distinguish similar-sounding actions ("fold" vs. "stir").

Moving beyond surface pattern matching requires architectures that reason about what *should* be present given procedural context. Several directions show promise:

(1) Training on execution traces. Pairing procedural text with execution logs—from cooking videos, robotic demonstrations, or simulated environments—makes implicit states explicit. This approach has shown encouraging results in robotics, where learning from demonstration provides supervision for unstated preconditions (Nair et al. 2021). Extending this paradigm to textual procedures could provide the missing signal for implicit state inference.

(2) Neuro-symbolic integration with procedural knowledge bases. Knowledge resources like VerbNet encode typical preconditions and effects for action classes. Early work integrating VerbNet for effect prediction (Clark et al. 2018) demonstrates that combining neural flexibility with such structured knowledge enables principled inference about unstated information. Following these encouraging results, future work could explore richer integration—for instance, using knowledge bases to generate candidate implicit states that neural models then verify against context.

(3) Self-supervised prediction of omissions. Training objectives that require predicting deliberately omitted steps or masked entity states could teach models to reason about expected-but-absent content. This approach mirrors successful masked language modeling but targets procedural structure rather than surface tokens. Related work in document understanding has shown that predicting elided content improves downstream reasoning (Czinczoll et al. 2024); adapting such objectives to procedural text is a concrete next step.

6.2 Maintaining Coherent World States

Explicitly representing and updating entity states instead of **treating procedures as flat token sequences**.
Addresses: State Maintenance (§5.3.2) → Entity Tracking, Robotics, Navigation

State tracking accuracy degrades sharply as procedures lengthen, even when the entire text fits within context limits (Kim and Schuster 2023; Mees et al. 2021). This pattern suggests that access to tokens alone is insufficient; standard architectures lack computational primitives for persistent state maintenance.

(1) External memory systems. Memory-augmented architectures such as Infini-attention and external associative-memory LLM (Das et al. 2024; Munkhdalai, Faruqui, and Gopal 2024) provide explicit read-write mechanisms that could store entity-attribute bindings updated by each action. While these architectures have shown promise in algorithmic tasks (Bulatov, Kuratov, and Burtsev 2022; Rodkin et al. 2024), their application to procedural state tracking remains underexplored. The challenge lies in learning *when* and *what* to write—decisions that require understanding how actions transform entity states.

(2) Dynamic structured representations. Current retrieval-augmented approaches treat structured data (scene graphs, entity databases) as static context to be retrieved, not dynamic state to be updated. This is insufficient for procedural understanding, where each step transforms the world. Future work should explore architectures where structured representations are *modified* by each procedural step: after "dice the onions," the entity representation for onions should update from `whole` to `diced`. Graph neural networks operating over entity-relation structures, updated incrementally as procedures unfold, offer one concrete instantiation of this idea.

(3) Verification through simulation and verifiable rewards. Recent work on inference-time scaling shows that allocating more computation at test time—e.g., sampling multiple solution paths and selecting via verifiers—can substantially improve performance on reasoning-heavy tasks (Snell et al. 2024; Liang et al. 2024). For procedural tasks, this additional computation could be used to verify state consistency: After each step, checking whether proposed state updates are coherent with prior states and action semantics. Reinforcement learning with verifiable rewards, where state consistency provides an automatically checkable training signal, is a promising paradigm. Such approaches can *mitigate* some forms of reward hacking by grounding learning in verifiable signals, but remain vulnerable to verifier/specification errors and verifier hacking, making verifier design critical.

The success of code-pretrained models on tracking tasks (Kim and Schuster 2023) provides an important clue: exposure to explicit variable assignment and state mutation during pretraining creates useful inductive biases. This suggests that procedural

corpora augmented with explicit state annotations, or training that incorporates code-like state operations, could substantially improve tracking capabilities.

6.3 Evaluating Procedural Understanding

Measuring reasoning validity instead of **rewarding output similarity**.
Addresses: Evaluation Gap (§5.3.3) → QA, Process Extraction, Summarization

A recurring pattern across procedural benchmarks is the disconnect between surface metrics and genuine understanding: High accuracy on end tasks coexists with poor performance on reasoning verification; strong token-level scores fail to predict whether outputs actually execute (Storks et al. 2021; Zhang et al. 2024a). This evaluation gap allows—and perhaps encourages—approaches that exploit surface shortcuts rather than developing genuine procedural competence.

(1) Intermediate state verification. Rather than evaluating only final outputs, metrics should require models to produce intermediate states verifiable against ground truth or simulation. For QA, this means evaluating the reasoning chain, not just the answer. For generation, this means checking that each step produces valid state transitions. The TRIP benchmark's tiered evaluation (Storks et al. 2021) provides a template: Separating task accuracy from reasoning validity reveals capabilities that aggregate metrics obscure.

(2) Robustness to meaning-preserving perturbations. If a system truly understands a procedure, its performance should be stable under surface variations that preserve procedural meaning—paraphrasing steps, reordering independent actions, substituting equivalent tools. Current benchmarks rarely test this robustness. Developing perturbation-based evaluation would help distinguish genuine understanding from surface pattern matching.

(3) Execution-based evaluation for ungrounded tasks. While grounded environments (Web, robotics, navigation) inherently test execution, this paradigm could extend to traditionally text-only tasks. Process extraction outputs could be executed in PDDL planners; generated procedures could be validated in lightweight simulators; summaries could be tested for whether they preserve executable structure. The key insight is that procedural validity is often *objectively verifiable*—unlike open-ended generation where quality is subjective. Building cheap-to-run procedural simulators that verify validity without requiring full embodied deployment could democratize execution-based evaluation and provide an automatic training signal.

6.4 Learning Hierarchical Abstractions

Decomposing procedures into goal hierarchies instead of **generating flat action sequences**.
Addresses: Compositional Generalization (§5.3.4) → Generation, Induction, Dialogue

Across procedural benchmarks, success on individual steps fails to transfer to step sequences—a pattern suggesting that tested approaches cannot organize procedures hierarchically (Zhang et al. 2020; Ye et al. 2025). Humans manage complex procedures through nested goal structures: a goal ("make dinner") decomposes into subgoals ("prepare vegetables," "cook protein," "make sauce"), each with its own sub-procedure

and completion criteria. Current approaches generate flat action sequences without this hierarchical scaffolding.

(1) Hierarchical generation with explicit goal decomposition. Rather than generating procedures end-to-end, approaches could first produce a goal hierarchy, then expand each subgoal into concrete steps. This mirrors how humans plan complex tasks and provides natural checkpoints for verification. Hierarchical reinforcement learning provides relevant technique: Options frameworks (Lin et al. 2024; Nayyar and Srivastava 2025) separate high-level goal selection from low-level action execution. Adapting these ideas to procedural text generation—where "options" correspond to subprocedures like "make the sauce" or "prepare the vegetables"—could enable compositional generalization that flat sequence models lack.

(2) Plan-then-execute with modular verification. Separating high-level planning from low-level generation allows each component to be trained and evaluated independently. Planning modules can be assessed on goal decomposition quality; execution modules on step-level correctness. This separation also enables hybrid approaches where planning uses symbolic methods (ensuring logical consistency) while execution uses neural generation (providing linguistic fluency). The key advantage is *modularity*: Failures can be localized to specific components, enabling targeted improvement.

(3) Curriculum learning with compositional structure. Training on short procedures before long ones, with explicit supervision on subgoal completion, could help models learn to compose rather than memorize. The critical design choice is curricula that increase *compositional depth* (more nested subgoals) rather than just sequence length. Work on over-generate-then-filter (Yuan et al. 2023) demonstrates that explicit verification at intermediate stages dramatically improves output quality—suggesting that learning to verify subgoal completion may be as important as learning to generate steps.

6.5 Grounding Language in Action

Learning from interactive execution instead of **text-only pretraining**.
Addresses: Grounding Gap (§5.3.5) → Web, Navigation, Robotics, GUI

Strong performance on text-only benchmarks does not transfer to action execution in interactive environments—substantial gaps persist across Web, navigation, and robotic settings (Zhou et al. 2023; Shridhar et al. 2020a; Padmakumar et al. 2022). Error analyses suggest failures are distributed across perception, navigation/action execution, and long-horizon planning/state modeling; improving perception alone helps substantially but does not solve the task (Ahn et al. 2022; Shi et al. 2024). This suggests that text-based training teaches associations between words and situations, but not the causal dynamics of action and effect.

(1) Pretraining on action-state transition data. Execution traces—logs that pair actions with resulting state changes—could provide the causal structure that text alone cannot. Sources include robotic demonstration datasets (Nair et al. 2021), gameplay recordings, and Web interaction logs. The challenge is scale: While text corpora contain billions of tokens, action-state data is orders of magnitude scarcer. Two paths forward are promising: simulation environments that generate unlimited interaction data (Makoviychuk et al. 2021; Shen et al. 2021), and learning from video where state changes are visible but actions must be inferred (Ko et al. 2023; Ye et al. 2024). Both

approaches have shown success in robotics and could extend to procedural language understanding.

(2) Active learning through environment interaction. Most procedural learning assumes passive observation of demonstration data. An underexplored alternative is active learning: systems propose actions, observe consequences, and update their understanding of action-state dynamics. This paradigm—central to reinforcement learning—enables learning directly from the environment's causal structure rather than from human annotations. Developing sample-efficient active learning methods for procedural grounding, where exploration is guided by uncertainty about action effects, is a key open direction.

(3) Multi-modal grounding beyond vision. Current grounding efforts focus heavily on visual input, but procedural execution involves richer sensory feedback: proprioception (knowing where your hands are), haptics (feeling resistance when cutting), and auditory cues (hearing when water boils). Robotic learning increasingly incorporates these modalities (Mees et al. 2021); extending multi-modal grounding to language understanding could enable more robust procedural execution. The finding that providing perfect visual information does not close the grounding gap (Shi et al. 2024) suggests that vision alone is insufficient—richer sensory grounding may be necessary for robust action execution.

6.6 Enabling Error Detection and Recovery

Detecting and correcting mistakes instead of **executing open-loop action sequences**.
Addresses: Error Accumulation (§5.3.6) → Web, Robotics, Dialogue

When procedures go wrong, errors compound: Each mistake corrupts the state for subsequent steps, and without detection mechanisms, systems continue executing on increasingly invalid assumptions (Kim, Baldi, and McAleer 2023; Zhou et al. 2023; Mees et al. 2021). Addressing this requires tackling two distinct but related problems.

The calibration problem. Systems cannot reliably distinguish what they know from what they do not know. Studies find high false-negative rates on task achievability judgments (Zhou et al. 2023) and systematic misclassification of infeasible tasks (Burns et al. 2022). This is fundamentally about self-knowledge: knowing when to proceed confidently versus when to seek clarification, try alternatives, or abandon a path. Improving calibration requires training signals that reward accurate uncertainty estimation, not just task success.

The recovery problem. Even with perfect uncertainty estimation, systems need mechanisms to recover from detected errors. Procedural domains vary in recoverability—clicking the wrong button on a Web site may be reversible; slicing an ingredient cannot be undone. Effective recovery requires understanding which actions are reversible, how to backtrack when possible, and how to find alternative paths when backtracking fails.

(1) Explicit progress monitoring. Systems should continuously compare expected states (predicted from the procedure) with observed states (from environment feedback). Discrepancies signal potential errors requiring investigation. This closed-loop architecture contrasts with current open-loop approaches that execute action sequences without verification. Implementing such monitoring connects directly to the state representation problem (§6.2): accurate expected-state prediction requires the coherent world models discussed there.

(2) Learning recovery policies from failure data. Beyond detecting errors, systems need strategies for responding. Recovery policies could be learned from human demonstrations of error correction, from simulation with injected failures, or from reinforcement learning with rewards for successful recovery. The robotics literature on fault-tolerant control provides relevant techniques (Kalithasan et al. 2024); adapting these to procedural language tasks—where "faults" are incorrect actions and "recovery" means replanning—is a concrete research direction.

(3) Separating uncertainty types. Current approaches conflate different uncertainty sources. Architectures that explicitly represent "I predict this action will fail" (epistemic uncertainty), "I don't know how to proceed" (capability uncertainty), and "this task cannot be done" (task constraint knowledge) could enable more appropriate responses: seeking information when uncertain about the world, trying alternatives when uncertain about approach, and communicating impossibility when task constraints preclude success.

6.7 Integrating Neural and Symbolic Methods

Combining learned flexibility with formal guarantees instead of **choosing one paradigm**.
Addresses: Multiple challenges, particularly Implicit Knowledge (§5.3.1) and Evaluation Gap (§5.3.3)

Neural approaches handle linguistic variation and data-driven generalization, while symbolic approaches enable interpretable reasoning and verification against formal constraints. Procedural understanding requires both: the flexibility to interpret diverse natural language instructions, and the rigor to ensure generated procedures are valid and executable. Studies showing that neural approaches capture action effects more reliably than preconditions (Zhang et al. 2024a) suggest they learn some procedural structure but miss formal constraints that symbolic methods could enforce.

(1) Neural proposal with symbolic verification. Neural models could generate candidate procedures or state updates that symbolic reasoners then verify for consistency. Invalid proposals trigger regeneration or refinement. This division of labor plays to each paradigm's strengths: Neural networks handle the ambiguity of natural language input; symbolic systems enforce logical constraints on outputs.

(2) Grounding in formal action schemas. PDDL and similar formalisms explicitly represent action preconditions, effects, and parameters. Neural approaches grounded in such schemas could learn to generate actions that respect formal constraints, with the schema providing automatic verification. Recent work on mapping open-domain procedural text to PDDL representations (Zhang et al. 2024a) provides a concrete bridge between natural language and formal planning models. The key advantage is that schema-grounded approaches can be trained with automatic supervision: Procedures that parse to valid PDDL and solve planning problems provide reward signal without human annotation.

(3) Differentiable symbolic modules. Rather than treating neural and symbolic components as separate systems in a pipeline, differentiable programming enables embedding *symbolic structure* within neural architectures—often via differentiable relaxations, solver-in-the-loop training, or neuralized operators—so parts of the system can be trained end-to-end. For procedural understanding, this could mean modules

that explicitly represent entities, apply structured state-update operations, and reason about preconditions, while learning parameters through gradient-based optimization.

(4) Tool use and function calling as applied grounding. The recent emergence of "tool use" and "function calling" capabilities in LLMs (Schick et al. 2023; Qin et al. 2024; Patil et al. 2024) represents a highly practical instantiation of neuro-symbolic integration. In this paradigm, the LLM serves as the neural reasoning engine that translates unstructured procedural intent into a formally-specified symbolic representation, typically a structured JSON schema. External tools and APIs then act as symbolic execution engines, directly instantiating the "neural proposal with symbolic verification" direction (Yao et al. 2023). Concretely, the model proposes a structured action, the API executes it or raises a schema validation error, and the neural model uses this deterministic feedback to iteratively correct its plan. Formalizing API schemas as modern equivalents to planning languages like PDDL offers a highly scalable pathway for grounding procedural language in executable actions (Qin et al. 2024), directly addressing the cross-domain fragmentation identified in Section 4.4. Open challenges remain, however, including hallucinated function calls, under-specified schemas, and the absence of precondition/effect validation mechanisms present in classical planning formalisms.

6.8 Improving Data Quality and Diversity

Curating procedurally-rich supervision instead of **scaling text quantity alone**.
Addresses: Multiple challenges; enables progress across all tasks

Current research concentrates heavily on WikiHow and recipe corpora, while other procedural domains remain underexplored. The finding that code-pretrained models outperform text-pretrained ones on state tracking (Kim and Schuster 2023) indicates that data composition—not just scale—shapes procedural capabilities. Several concrete directions could diversify and enrich procedural training data:

(1) Domain expansion to technical and multilingual sources. Troubleshooting documentation from technology companies, repair manuals (iFixit), and industrial process descriptions offer procedural text with characteristics absent from cooking: more implicit domain knowledge, specialized terminology, longer dependency chains, and different failure modes. These resources remain largely untapped, partly due to inconsistent formatting that complicates extraction. Building standardized pipelines—perhaps using LLMs for initial parsing followed by human verification—could unlock this diversity.

Similarly, non-English procedural resources are underexplored: recipe sites in Turkish, Chinese, or Hindi; WikiHow's multilingual editions; local how-to forums. These offer both cross-lingual training data and evaluation benchmarks for testing whether procedural capabilities transfer across languages.

(2) Execution trace alignment for implicit state annotation. Current corpora annotate surface text but rarely capture the implicit states that procedures assume. Execution traces—from cooking videos, robotic demonstrations, or game playthroughs—make these states observable. For example, a cooking video reveals that "sauté until soft" involves visible texture and color changes that text leaves implicit; a robotic demonstration shows the gripper configurations and force patterns that "pick up the cup" requires.

Aligning such traces with procedural text is labor-intensive but feasible. Existing resources provide starting points: EPIC-KITCHENS (Damen et al. 2021) pairs cooking

videos with action annotations; robotic demonstration datasets (Nair et al. 2021) include state observations; game replay databases record action-state sequences. The research challenge is developing alignment methods that map continuous sensory traces to discrete textual descriptions—a form of grounded language learning that could provide supervision unavailable from text alone.

(3) Hierarchical annotation for long-horizon procedures. Current datasets typically contain 5–12 steps, yet real-world procedures often span dozens or hundreds of steps with nested subprocedures. WikiHow's method-step-substep structure provides some hierarchy, but most datasets flatten this into linear sequences, losing the goal structure that enables human comprehension of complex procedures.

Creating datasets that preserve hierarchical structure—explicitly marking which steps serve which subgoals, which subgoals combine into which higher goals—would enable training and evaluation of hierarchical approaches (§6.4). The business process modeling literature, with BPMN representations of complex workflows (Qian et al. 2020), offers annotation schemas for such hierarchy. Adapting these schemas to natural language procedures, perhaps through crowdsourced annotation of existing corpora, could provide the structured supervision that flat sequence data cannot.

(4) Validated synthetic data through execution filtering. Large language models can generate procedural text at scale, but such text often contains subtle errors: invalid action sequences, impossible state transitions, physically implausible steps. CoScript (Yuan et al. 2023) demonstrates an over-generate-then-filter paradigm using constraint-faithfulness filtering; extending this to execution-based filtering in simulators is a promising next step.

(5) Paraphrasing for linguistic diversity. Synthetic datasets offer control over procedural complexity but typically lack linguistic diversity, using templated language that fails to capture natural variation. A promising direction, successfully applied in other NLP domains, is two-stage generation: First create synthetic procedures for their structural properties (correct action sequences, valid state transitions), then crowdsource paraphrases to add linguistic naturalism (Kim and Schuster 2023). This combination yields datasets that are both structurally controlled—enabling systematic evaluation of procedural capabilities—and linguistically natural—enabling transfer to real-world text.

7. Conclusion

This survey provides a systematic analysis of procedural language understanding across representation schemes, resources, and downstream tasks. Examining 181 papers, we identify fundamental patterns, persistent challenges, and opportunities for future work.

Our analysis reveals a central tension between current representation paradigms: Event-centric approaches often capture inter-step dependencies but under-specify entity state evolution, while entity-centric representations track state changes but often treat actions as opaque triggers. Unified representations that jointly model both aspects remain rare. We observe clear patterns in how different representations serve different tasks, yet also find notable gaps where representation–task combinations remain underexplored.

Benchmark performance often disconnects from real-world applicability. Models achieving high scores on automatic metrics can still fail when their outputs are executed, revealing fundamental evaluation limitations. Six core challenges persist across tasks:

Implicit knowledge (crucial information is unstated), compounding failures in state maintenance, evaluation frameworks that reward surface similarity over functional correctness, compositional reasoning failures, the gap between language understanding and grounded action, and the inability to detect and recover from errors.

The field has evolved from isolated event extraction and entity tracking toward integrated procedural understanding. As large language models demonstrate instruction-following capabilities in short-horizon settings, research increasingly addresses complex, multi-step procedures requiring simultaneous event understanding, entity tracking, and environmental grounding. Continued progress necessitates advances in evaluation methodology, data curation, and architectural innovations—including neuro-symbolic integration, hierarchical planning, and error recovery mechanisms.

Appendix A. Database Specific Keyword Settings

The query includes 13 sub-queries focusing on frequent terms in procedural text research. Initially, we used broad keywords like "procedural text" and "instructional text," then refined the search to cover common corpora, instruction forms, and tasks. The full list is given in Table A.1.

Table A.1
Keywords.

Google Scholar, IEEE Xplore	DBLP
Exact Phrase Match (PM)	Exact Match per Word (EMW)
"procedural text"	
"instructional text"	
"natural language instruction(s)"	
"entity tracking"	
"entity state tracking"	
"wikihow"	
"script knowledge"	
"BPMN generation"	
"action sequence natural language"	
"action sequence annotation"	
"instruction parsing"	
"instruction manual" AND "parsing"	
"recipe parsing"	

Appendix B. Grounded Task-Oriented Dialogue Datasets

Table B.1 showcases a variety of datasets used in grounded task-oriented dialogue research, illustrating diverse tasks ranging from single subtasks, such as Knowledge Identification (which can be mentioned with different names, including User Utterance Grounding, Grounding Span Prediction, Flow-chart Retrieval) to full pipeline tasks like Cascade Dialogue Success, and using different metrics.

Table B.1
Commonly used grounded task-oriented dialogue datasets.

Dataset	Docs	Dialogues	Evaluation			
			Task	Metric	Baseline	SoTA
Doc2Dial	480	4,470	Knowledge Identification	EM F1	55.4 66.6	68.4 (Daheim et al. 2021) 88.7 (Daheim et al. 2022)
MultiDoc2Dial	488	4,796	Knowledge Identification	EM F1	26.6 43.7	51.0 (Zhao et al. 2023) 64.5 (Zhao et al. 2023)
Task2Dial	353	478	–	–	–	–
CookDial	260	260	User Question Understanding	Accuracy F1	94.5 91.0	– –
ABCD	55	10,042	Cascade Dialgoue Success	Cascading Eval	31.9	60.7 (Ramakrishnan et al. 2023)
FLODial	12	5,476	Knowledge Identification	Success Rate R@1	33.7 91.0	67.1 (Hu et al. 2024) –
TEACh	12	3,047	Execution from Dialogue History	SR GC	7.06 9.57	18.60 (Jain et al. 2024) 18.60 (Jain et al. 2024)
			Trajectory from Dialogue	SR GC	0.51 20.3	– –
			Two-Agent Task Completion	SR	24.4	–

Appendix C. List of Grounded and Ungrounded Task Papers

List of all papers for grounded and ungrounded tasks are given in Table C.1.

Table C.1
Ungrounded and grounded tasks papers.

Category	Task/Environment	Papers
Ungrounded	Summarization	(Ladhak et al. 2020; Koupaee and Wang 2018; Le and Tuan 2024; Boni et al. 2021; DeChant and Bauer 2022; Shridhar et al. 2020a; Scialom et al. 2019; Le and Tuan 2024)
	Event Alignment	(Lin et al. 2020; Nandy, Kapadnis et al. 2023; Regneri et al. 2010; Wanzare et al. 2017; Donatelli et al. 2021; Marin et al. 2019; Bień et al. 2020; Wanzare et al. 2016; Mavroudi, Afouras, and Torresani 2023; Krishna et al. 2017; Shridhar et al. 2020a; Scialom et al. 2019; Dalvi et al. 2018; Zhou, Xu, and Corso 2018; Kwon and Lee 2025)
	Detecting and Correcting Implicit Instructions	(Anthonio, Bhat, and Roth 2020; Roth and Anthonio 2021; Zeng et al. 2020; Zhang, Yamakata, and Tajima 2021; Bisk et al. 2019; Feng et al. 2022; Clark et al. 2018; Cheng and Erk 2018; Zeng et al. 2020; Zhang, Wu, and Cai 2023)
	Entity State Tracking	(Tandon et al. 2020; Dalvi et al. 2019; Goyal et al. 2021; Zhang et al. 2024; Rim et al. 2023; Dalvi et al. 2018; Clark et al. 2018; Zhang et al. 2021b; Long, Pasupat, and Liang 2016; Li, Nye, and Andreas 2021; Kim and Schuster 2023; Bosselut et al. 2018; Zhang et al. 2023b; Cheng and Erk 2018; Kazeminejad and Palmer 2023; Nandy, Kapadnis et al. 2023; Nandy, Kulkarni et al. 2024; Dalvi et al. 2018; Chen et al. 2019)
	Instruction Parsing	(Harashima and Hiramatsu 2020; Pan et al. 2020; Bhatt et al. 2024; Zhang, Yamakata, and Tajima 2022; Costa et al. 2017; Nabizadeh, Kolossa, and Heckmann 2020; Miglani and Yorke-Smith 2020; Zhang et al. 2021a; Feng, Zhuo, and Kambhampati 2018; Maeta, Sasada, and Mori 2015; Park and Motahari-Nezhad 2018; Mysore et al. 2019; Qian et al. 2020; Diwan, Batra, and Bagler 2020; Ito, Suzuki, and Aizawa 2020; Ri et al. 2022; Zhou et al. 2022; Rula and D'Souza 2023; Wang et al. 2022; Kourani et al. 2024; Kumaran et al. 2024; Regneri et al. 2010; Wanzare et al. 2016, 2017; Liu et al. 2018; Zhang et al. 2020; Wu et al. 2023a; Fan and Hunter 2023; Bellan et al. 2023; Lal et al. 2024; Zhang et al. 2024, 2024a; Miech et al. 2019; Kiddon et al. 2015; Regneri et al. 2010; Wanzare et al. 2016, 2017; Liu et al. 2018; Shi et al. 2017; Batra et al. 2020; Salvador et al. 2017; Brach, Košťál, and Ries 2025)
	Process Generation	(Lyu, Zhang, and Callison-Burch 2021; Zhang et al. 2020; Sun et al. 2023; Kiddon, Zettlemoyer, and Choi 2016; Rojowiec et al. 2020; Liu et al. 2022; Lu et al. 2023; Nishimura et al. 2021; Modi et al. 2016; Sakaguchi et al. 2021; Nguyen et al. 2017; Yuan et al. 2023; Wanzare et al. 2016; Sakaguchi et al. 2021; Li et al. 2025)
	Question Answering	(Bolotova-Baranova et al. 2023; Zhou, Shah, and Schockaert 2019; Uzunoglu, Şahin, and Safa 2024; Zhang, Lyu, and Callison-Burch 2020; Yang et al. 2021; Ostermann et al. 2018; Ostermann, Roth, and Pinkal 2019; Zellers et al. 2019; Storks et al. 2021; Yagcioglu et al. 2018; Bauer and Bansal 2021; Weston et al. 2016; Tandon, Dalvi et al. 2019; Wang et al. 2023; Zhang, Lu, and Jaitly 2024)
	Knowledge Acquisition/ Mining	(Chen et al. 2020; Jung et al. 2010; Chu et al. 2017; Steinert and Meneguzzi 2020; Sen et al. 2023; Miglani and Yorke-Smith 2020; Babli and Onaindía 2019)
Grounded	Web	(Gur et al. 2024; Mazumder and Riva 2021; Xu et al. 2021; Kim, Baldi, and McAleer 2023; Deng et al. 2023; Liu et al. 2018; Zhou et al. 2023; Nakano et al. 2021; Yao et al. 2022; Shi et al. 2017; Deng et al. 2024; Zhu et al. 2024)
	Simulations/Navigation	(Wu et al. 2024; Rajvanshi et al. 2023; Das et al. 2017; Misra et al. 2015; Li et al. 2024; Zhou, Yin, and Neubig 2022; Vogel and Jurafsky 2010; Misra et al. 2018; Li et al. 2022c; Qi et al. 2020; Misra et al. 2014; Zang et al. 2018; Feng et al. 2022; Puig et al. 2018; Anderson et al. 2017; MacMahon, Stankiewicz, and Kuipers 2006; Artzi and Zettlemoyer 2013; Shi et al. 2024)
	Robotic	(Howard, Tellex, and Roy 2014; Tellex et al. 2011a; Mees et al. 2021; Tellex et al. 2011b; Nair et al. 2021; Singh et al. 2022; Bisk, Yuret, and Marcu 2016; Scalise et al. 2018; Bucker et al. 2022; Ahn et al. 2022; Matuszek et al. 2012; Cleveston et al. 2025)
	Game	(Toshniwal et al. 2021; Narayan-Chen, Jayannavar, and Hockenmaier 2019; Lachmy et al. 2022; Li et al. 2022a; Hu et al. 2019; Jayannavar, Narayan-Chen, and Hockenmaier 2020; Joshi et al. 2023; Wu et al. 2023b; Marc-Alex et al. 2018; Hooshyar et al. 2018)
	GUI/Mobile	(Burns et al. 2022; Toyama et al. 2021; Banerjee et al. 2023; Li et al. 2020; Zhang et al. 2023a
	Dialogue	(Chen et al. 2021; Shao and Nakashole 2020; Narayan-Chen, Jayannavar, and Hockenmaier 2019; Jiang et al. 2022; Srinet et al. 2020; Gao et al. 2022; Feng et al. 2020; Raghu et al. 2021; Moghe et al. 2024; Sun et al. 2022; Ammanabrolu and Riedl 2021; Feng et al. 2021; Strathearn and Gkatzia 2022; Padmakumar et al. 2022)

Appendix D. Grounded Environment Details

We have identified the following divergences across environments: *data instance* (typical components of the dataset), *dataset size*, *the action space*, *tasks performed* in the *environment*. The end tasks vary depending on the environment. For instance, simulating user actions on Web sites, such as filling forms or navigating menus (Zhou et al. 2023; Deng et al. 2023; Yao et al. 2022) are common in Web environments, while gaming scenarios often involve character navigation and object interaction within virtual worlds (Wu et al. 2023b; Narayan-Chen, Jayannavar, and Hockenmaier 2019; Wu et al. 2023b; Lachmy et al. 2022). A summary is given in Table D.1.

Table D.1
Grounded instructions. Screen: The structured UI state/tree not pixels.

	Environment	Data instance	Task	# of Actions	Size
PixelHelp (Li et al. 2020)	Google Phone	Screen, instruction, grounded UI actions	Account configuration, Gmail, Chrome, Photos tasks	2–8 steps	187
WinHelp (Branavan et al. 2009)	Windows 2000 VM	Screen, instruction, grounded UI actions	Windows troubleshooting	10,3	128
AndroidHowTo (Li et al. 2020)	Google Phone	Screen, instruction, grounded UI actions	Account configuration, Gmail, Chrome, Photos tasks	2–8 steps	187
AndroidEnv (Toyama et al. 2021)	Mobile	Emulated Android devices, touchscreen gestures, real-time interaction	UI navigation, game playing, and basic utilities	Varies	100 (tasks)
WebArena (Zhou et al. 2023)	Web	Screenshot/HTML Dom Tree/Accessibility Tree, Multitab, (high-level) instruction, grounded UI actions	Web-based tasks for e-commerce, social forums, collaborative software, and content management	Varies	812
Mind2Web (Deng et al. 2023)	Web	Real-world Web sites, instructions, and action sequences	variety of tasks across different domains on any Web site	Average = 7.3	2,350 (from 137 Web sites across 31 domains)
World of Bits (Shi et al. 2017)	Web	Screen pixels, DOM, keyboard and mouse actions	Web-based tasks via natural language queries	Varies	100 (Web tasks)
WOB++ (Liu et al. 2018)	Web	DOM elements, workflow steps (Click, Type)	Email processing, form filling, etc., in noisy environments	Varies	80 tasks
CrossBlock (Liu et al. 2018)	Grid	Grid state, instruction, action	Clear the grid by drawing lines	5,86	50
Hexagons (Draw me a flower) (Lachmy et al. 2022)	Grid	Grid state, instruction, action	Fill the grids with colors	6,73	620
Chess (Toshniwal et al. 2021)	Grid	Chessboard state, instructions	Predicting legal chess moves	Varies	2.5M scenarios
SCONE (Long, Pasupat, and Liang 2016)	Simulation	Instructions, Logical form	Infer the final environment state following the actions	5	4K
VirtualHome (Puig et al. 2018)	3D Simulation	Instructions, Scratch Program, Video	Household	11,6	2,821
LANI, CHAI (Misra et al. 2018)	3D World w/Landmarks	Instructions, world states, actions	Navigation/Household	4,7/7,7	6,000/1,596
CALVIN (Mees et al. 2021)	Robotics/Simulation	language instructions, multimodal sensors	Long-horizon tasks (e.g., open drawer, push block)	∼5–10	20K language directives
Bucker et al. (2022)	Robotics/Simulation	Predefined trajectories, object positions, natural language commands	Trajectory reshaping based on natural language commands	∼5–10 steps	∼10,000 trajectory labels
LOReL (Nair et al. 2021)	Robotics/Simulation	Vision-based interaction, natural language commands, multimodal sensors	Language-conditioned manipulation (e.g., open drawer, move stapler)	up to 150	53,000 scenarios
Meta-GUI (Sun et al. 2022)	Dialogue/Web	Screenshots, HTML Dom Tree/Accessibility Tree, touch and text inputs	Multi-modal conversational interactions (e.g., booking a hotel, checking the weather)	∼4–5	1,125 dialogues

Appendix E. Example of Event-Centric Representations

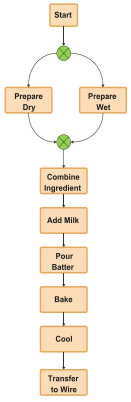
Figure E.1 illustrates the diversity of structures used to represent the process of baking a cake through three different formalisms. While all three approaches capture the same goal, the way in which they structure the actions and entities involved is distinct. Figure E.2 shows an example of BPMN for a "Sending an issue list" process.

Ingredients:
1- 1 ½ cups all-purpose flour
2- 1 ½ teaspoons baking powder
3- ½ teaspoon salt
4- ½ cup unsalted butter, softened
5- 1 cup granulated sugar
6- 2 large eggs
7- 1 teaspoon vanilla extract
8- ½ cup milk
Instructions:
1. Prepare dry ingredients: Mix flour, baking powder, and salt.
2. Prepare wet ingredients: Cream butter and sugar, add eggs, and vanilla.
3. Combine ingredients: Gradually add dry ingredients to wet ingredients while mixing.
4. Add Milk.
5. Pour batter into a greased 8-inch cake pan.
6. Bake for 25-30 minutes at 350°F (175°C).
7. Let cool for 10 minutes
8. Transfer to a wire rack.

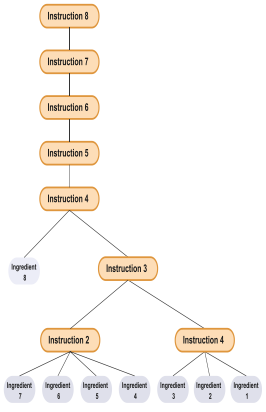
(a) Recipe Text

Prepare (1- $1\frac{1}{2}$ cups all-purpose flour, $1\frac{1}{2}$ teaspoons baking powder, $\frac{1}{2}$ teaspoon salt)
→
Prepare ($\frac{1}{2}$ cup unsalted butter, softened, 1 cup granulated sugar, 2 large eggs, 1 teaspoon vanilla extract)
→
Combine (ingredients)
→
Add ($\frac{1}{2}$ cup milk)
→
Pour (batter, into greased 8-inch cake pan)
→
Bake (25-30 minutes at 350°F)
→
Cool (10 minutes)
→
Transfer (to wire rack)

(b) Recipe Representation in Wiki HG (Feng, Zhuo, and Kambhampati 2018)



(c) Recipe Representation as Process Graph (Qian et al. 2020)



(d) Recipe Representation in SIMMR (Jermsurawong and Habash 2015)

Figure E.1
Vanilla cake recipe representations: (a) Recipe text, (b) text representation from Wiki HG (Feng, Zhuo, and Kambhampati 2018), (c) Process graph (Qian et al. 2020), and (d) SIMMR structure (Jermsurawong and Habash 2015).

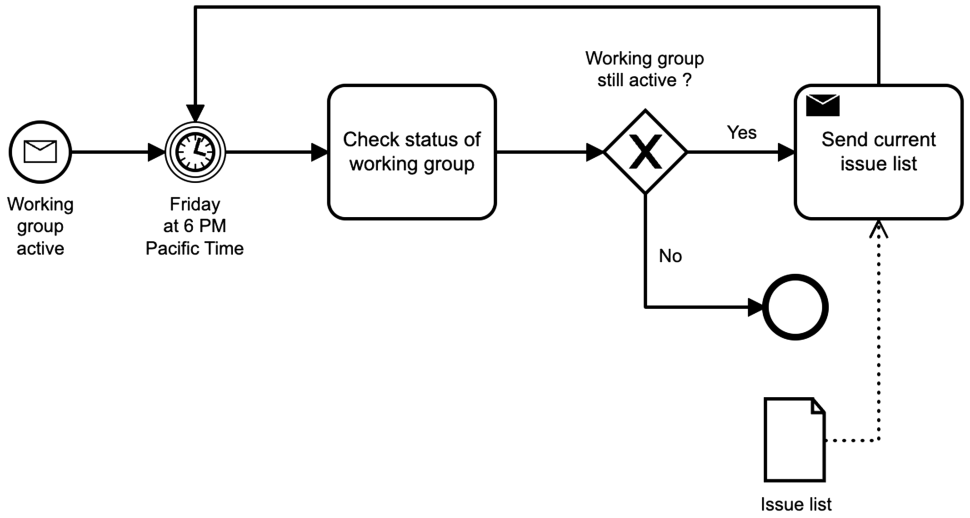


Figure E.2
Example of a business process model and notation for "Sending an issue list" process (Wikipedia 2024).

Acknowledgments

This work has been supported by the Scientific and Technological Research Council of Türkiye (TÜBİTAK) as part of the project "Automatic Learning of Procedural Language from Natural Language Instructions for Intelligent Assistance" with the number 121C132. We also gratefully acknowledge KUIS AI Lab for providing computational support. Special thanks to Müge Kural for her valuable assistance during the paper identification stage.

References

Ahn, Michael, Anthony Brohan, Noah Brown, et al. 2022. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning*.

Ammanabrolu, Prithviraj and Mark O. Riedl. 2021. Modeling worlds in text. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021*.

Anderson, Peter, Qi Wu, Damien Teney, et al. 2017. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3674–3683. https://doi.org/10.1109/CVPR.2018.00387

Anthonio, Talita, Irshad Bhat, and Michael Roth. 2020. WikiHowToImprove: A resource and analyses on edits in instructional texts. In *Proceedings of the 12th Language Resources and Evaluation Conference, LREC 2020*, pages 5721–5729.

Artzi, Yoav and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62. https://doi.org/10.1162/tacl_a_00209

Babli, Mohannad and Eva Onaindía. 2019. A context-aware knowledge acquisition for planning applications using ontologies. *arXiv:1904.09845*.

Banerjee, Pratyay, Shweti Mahajan, Kushal Arora, et al. 2023. Lexi: Self-supervised learning of the UI language. *arXiv:2301.10165*.

Batra, Devansh, Nirav Diwan, Utkarsh Upadhyay, et al. 2020. RecipeDB: A resource for exploring recipes. *Database*, 2020:baaa077. https://doi.org/10.1093/database/baaa077, PubMed: 33238002

Bauer, Lisa and Mohit Bansal. 2021. Identify, align, and integrate: Matching knowledge graphs to commonsense reasoning tasks. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2259–2272. https://doi.org/10.18653/v1/2021.eacl-main.192

Bellan, Patrizio, Han van der Aa, Mauro Dragoni, et al. 2023. PET: An annotated dataset for process extraction from natural language text tasks. In *Business Process Management Workshops*, pages 315–321. https://doi.org/10.1007/978-3-031-25383-6_23

Bhatt, Dhaivat J., Seyed Ahmad Abdollahpouri Hosseini, Federico Fancellu, et al. 2024. End-to-end parsing of procedural text into flow graphs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5833–5842. https://doi.org/10.63317/3p66g4asgi95

Bień, Michał, Michał Gilski, Martyna Maciejewska, et al. 2020. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 22–28. https://doi.org/10.18653/v1/2020.inlg-1.4

Bisk, Yonatan, Jan Buys, Karl Pichotta, et al. 2019. Benchmarking hierarchical script knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4077–4085. https://doi.org/10.18653/v1/N19-1412

Bisk, Yonatan, Deniz Yuret, and Daniel Marcu. 2016. 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*, pages 751–761. https://doi.org/10.18653/v1/N16-1089

Bolotova-Baranova, Valeria, Vladislav Blinov, Sofya Filippova, et al. 2023. WikiHowQA: A comprehensive benchmark for multi-document non-factoid question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, pages 5291–5314. https://doi.org/10.18653/v1/2023.acl-long.290

Bombieri, Marco, Daniele Meli, Diego Dall'Alba, et al. 2023. Mapping natural

language procedures descriptions to linear temporal logic templates: An application in the surgical robotic domain. *Applied Intelligence*, 53(22):26351–26363. https://doi.org/10.1007/s10489-023-04882-0

Boni, Odellia, Guy Feigenblat, Guy Lev, et al. 2021. HowSumm: A multi-document summarization dataset derived from WikiHow articles. *arXiv:2110.03179*.

Bosselut, Antoine, Omer Levy, Ari Holtzman, et al. 2018. Simulating action dynamics with neural process networks. In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*, OpenReview.net.

Brach, William, Kristián Košťál, and Michal Ries. 2025. The effectiveness of large language models in transforming unstructured text to standardized formats. *arXiv:2503.02650*. https://doi.org/10.1109/ACCESS.2025.3573030

Branavan, S. R. K., Harr Chen, Luke Zettlemoyer, et al. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90. https://doi.org/10.3115/1687878.1687892

Breit, Anna, Laura Waltersdorfer, Fajar J. Ekaputra, et al. 2023. Combining machine learning and semantic Web: A systematic mapping study. *ACM Computing Surveys*, 55:1–41. https://doi.org/10.1145/3586163

Bucker, Arthur Fender C., Luis F. C. Figueredo, Sami Haddadin, et al. 2022. Reshaping robot trajectories using natural language commands: A study of multi-modal data alignment using transformers. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 978–984. https://doi.org/10.1109/IROS47612.2022.9981810

Bulatov, Aydar, Yuri Kuratov, and Mikhail S. Burtsev. 2022. Recurrent memory transformer. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22. https://doi.org/10.52202/068431-0805

Burns, Andrea, Deniz Arsan, Sanjna Agrawal, et al. 2022. A dataset for interactive vision-language navigation with unknown command feasibility. In *Computer Vision - ECCV 2022 - 17th European Conference, Proceedings, Part VIII*, volume 13668 of *Lecture Notes in Computer Science*, pages 312–328. https://doi.org/10.1007/978-3-031-20074-8_18

Chandu, Khyathi Raghavi, Yonatan Bisk, and Alan W. Black. 2021. Grounding 'grounding' in NLP. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021*, pages 4283–4305. https://doi.org/10.18653/v1/2021.findings-acl.375

Chen, Derek, Howard Chen, Yi Yang, et al. 2021. Action-based conversations dataset: A corpus for building more in-depth task-oriented dialogue systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, pages 3002–3017. https://doi.org/10.18653/v1/2021.naacl-main.239

Chen, Guangyong, Pengfei Chen, Chang-Yu Hsieh, et al. 2019. Alchemy: A quantum chemistry dataset for benchmarking AI models. *arXiv:1906.09427*.

Chen, Hao, Abdul Waheed, Xiang Li, Yidong Wang, Jindong Wang, Bhiksha Raj, and Marah I. Abdin. 2024. On the diversity of synthetic data and its impact on training large language models. *arXiv:2410.15226*.

Chen, Huiyao, Meishan Zhang, Jing Li, Min Zhang, Lilja Øvrelid, Jan Hajič, and Hao Fei. 2025. Semantic role labeling: A systematical survey. *arXiv preprint arXiv:2502.08660*.

Chen, Muhao, Hongming Zhang, Qiang Ning, et al. 2021b. Event-centric natural language processing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Tutorial Abstracts*, pages 6–14. https://doi.org/10.18653/v1/2021.acl-tutorials.2

Chen, Muhao, Hongming Zhang, Haoyu Wang, et al. 2020. What are you trying to do? Semantic typing of event processes. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 531–542. https://doi.org/10.18653/v1/2020.conll-1.43

Cheng, Pengxiang and Katrin Erk. 2018. Implicit argument prediction with event knowledge. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 831–840. https://doi.org/10.18653/v1/N18-1076

Chu, Cuong Xuan, Niket T., on, et al. 2017. Distilling task knowledge from how-to

communities. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017*, pages 805–814. https://doi.org/10.1145/3038912.3052715

Clark, Peter, Bhavana Dalvi, Niket T., et al. 2018. What happened? Leveraging VerbNet to predict the effects of actions in procedural text. *arXiv:1804.05435*.

Cleveston, Iury, Alana de Santana Correia, Paula D. P. Costa, Ricardo R. Gudwin, Alexandre da Silva Simões, and Esther Luna Colombini. 2025. InstructRobot: A model-free framework for mapping natural language instructions into robot motion. *CoRR*, abs/2502.12861.

Cohen, Vanya, Jason Xinyu Liu, Raymond Mooney, et al. 2024. A survey of robotic language grounding: Tradeoffs between symbols and embeddings. *arXiv:2405.13245*.

Costa, Carlos M., Germano Veiga, Arm Sousa, et al. 2017. Evaluation of Stanford NER for extraction of assembly information from instruction manuals. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 302–309. https://doi.org/10.1109/ICARSC.2017.7964092

Czinczoll, Tamara, Christoph Hönes, Maximilian Schall, and Gerard De Melo. 2024. NextLevelBERT: Masked language modeling with higher-level representations for long documents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4656–4666. https://doi.org/10.18653/v1/2024.acl-long.256

Dagan, Gautier, Frank Keller, and Alex Lascarides. 2025. Plancraft: An evaluation dataset for planning with LLM agents. *arXiv:2412.21033*.

Daheim, Nico, David Thulke, Christian Dugast, et al. 2021. Cascaded span extraction and response generation for document-grounded dialog. In *Proceedings of the 1st Workshop on Document-grounded Dialogue and Conversational Question Answering (DialDoc 2021)*, pages 57–62. https://doi.org/10.18653/v1/2021.dialdoc-1.8

Daheim, Nico, David Thulke, Christian Dugast, et al. 2022. Controllable factuality in document-grounded dialog systems using a noisy channel model. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1365–1381. https://doi.org/10.18653/v1/2022.findings-emnlp.98

Dalvi, Bhavana, Lifu Huang, Niket T., et al. 2018. Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, Volume 1 (Long Papers)*, pages 1595–1604. https://doi.org/10.18653/v1/N18-1144

Dalvi, Bhavana, Niket Tandon, Antoine Bosselut, Wen-tau Yih, and Peter Clark. 2019. Everything happens for a reason: Discovering the purpose of actions in procedural text. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pages 4495–4504. https://doi.org/10.18653/v1/D19-1457

Damen, Dima, Hazel Doughty, Giovanni Maria Farinella, et al. 2021. The EPIC-KITCHENS dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 43(11):4125–4141. https://doi.org/10.1109/TPAMI.2020.2991965, PubMed: 32365017

Das, Abhishek, Samyak Datta, Georgia Gkioxari, et al. 2017. Embodied question answering. *arXiv:1711.11543*.

Das, Payel, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarathkrishna Swaminathan, Sihui Dai, Aurélie Lozano, Georgios Kollias, Vijil Chenthamarakshan, Jiří Navrátil, Soham Dan, and Pin-Yu Chen. 2024. Larimar: Large language models with episodic memory control. In *Proceedings of ICML'24*, JMLR.org.

DeChant, Chad and Daniel Bauer. 2022. Summarizing a virtual robot's past actions in natural language. *arXiv:2203.06671*.

Deng, Xiang, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a generalist agent for the Web. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.

Deng, Yang, Xuan Zhang, Wenxuan Zhang, et al. 2024. On the multi-turn instruction following for conversational Web agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8795–8812. https://doi.org/10.18653/v1/2024.acl-long.477

Diwan, Nirav, Devansh Batra, and Ganesh Bagler. 2020. A named entity based approach to model recipes. In *2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW)*, pages 88–93. https://doi.org/10.1109/ICDEW49219.2020.000-2

Donatelli, Lucia, Theresa Schmidt, Debanjali Biswas, et al. 2021. Aligning actions across recipe graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 6930–6942. https://doi.org/10.18653/v1/2021.emnlp-main.554

Fagnou, Erwan, Paul Caillon, Blaise Delattre, and Alexandre Allauzen. 2024. Chain and causal attention for efficient entity tracking. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13174–13188. https://doi.org/10.18653/v1/2024.emnlp-main.731

Fan, Yi and Anthony Hunter. 2023. Understanding the cooking process with English recipe text. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4244–4264. https://doi.org/10.18653/v1/2023.findings-acl.261

Feng, Song, Siva Sankalp Patel, Hui Wan, et al. 2021. MultiDoc2Dial: Modeling dialogues grounded in multiple documents. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/2021.emnlp-main.498

Feng, Song, Hui Wan, Chulaka Gunasekara, et al. 2020. Doc2Dial: A goal-oriented document-grounded dialogue dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8118–8128. https://doi.org/10.18653/v1/2020.emnlp-main.652

Feng, Weixi, Tsu-Jui Fu, Yujie Lu, et al. 2022. ULN: Towards underspecified vision-and-language navigation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 6394–6412. https://doi.org/10.18653/v1/2022.emnlp-main.429

Feng, Wenfeng, Hankz Hankui Zhuo, and Subbarao Kambhampati. 2018. Extracting action sequences from texts based on deep reinforcement learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 4064–4070, ijcai.org. https://doi.org/10.24963/ijcai.2018/565

Gao, Xiaofeng, Qiaozi Gao, Ran Gong, et al. 2022. DialFRED: Dialogue-enabled agents for embodied instruction following. *IEEE Robotics and Automation Letters*, 7:10049–10056. https://doi.org/10.1109/LRA.2022.3193254

Ghazarian, Sarik, Abhinav Gullapalli, Swair Shah, Anurag Beniwal, Nanyun Peng, Narayanan Sadagopan, and Zhou Yu. 2025. TOD-ProcBench: Benchmarking complex instruction-following in task-oriented dialogues. *arXiv preprint arXiv:2511.15976*.

Gonzalez-Pumariega, Gonzalo, Leong Su Yean, Neha Sunkara, and Sanjiban Choudhury. 2025. Robotouille: An asynchronous planning benchmark for LLM agents. *arXiv:2502.05227*.

Goyal, Saransh, Pratyush Pandey, et al. 2021. Tracking entities in technical procedures—A new dataset and baselines. *arXiv:2104.07378*.

Guo, Daya, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Guo, Yanzhu, Guokan Shang, Michalis Vazirgiannis, and Chloé Clavel. 2024. The curious decline of linguistic diversity: Training language models on synthetic text. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3589–3604. https://doi.org/10.18653/v1/2024.findings-naacl.228

Gur, Izzeddin, Hiroki Furuta, Austin Huang, et al. 2024. A real-world WebAgent with planning, long context understanding, and program synthesis. *arXiv:2307.12856*.

Harashima, Jun and Makoto Hiramatsu. 2020. Cookpad Parsed Corpus: Linguistic annotations of Japanese recipes. In *Proceedings of the 14th Linguistic Annotation Workshop*, pages 87–92.

He, Hongliang, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024a. WebVoyager: Building an end-to-end Web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890. https://doi.org/10.18653/v1/2024.acl-long.371

He, Qianyu, Jie Zeng, Wenhao Huang, et al. 2024b. Can large language models understand real-world complex instructions? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):18188–18196. https://doi.org/10.1609/aaai.v38i16.29777

Hendrycks, Dan, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *arXiv:2009.03300*.

Hendrycks, Dan, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1 (NeurIPS Datasets and Benchmarks 2021)*.

Hooshyar, Danial, Moslem Yousefi, Minhong Wang, et al. 2018. A data-driven procedural-content-generation approach for educational games. *Journal of Computer Assisted Learning*, 34:731–739. https://doi.org/10.1111/jcal.12280

Howard, Thomas M., Stefanie Tellex, and Nicholas Roy. 2014. A natural language planner interface for mobile manipulators. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6652–6659. https://doi.org/10.1109/ICRA.2014.6907841

Hu, Hengyuan, Denis Yarats, Qucheng Gong, et al. 2019. Hierarchical decision making by generating and following natural language instructions. *arXiv:1906.00744*.

Hu, Zhiyuan, Chumin Liu, Xidong Feng, et al. 2024. Uncertainty of thoughts: Uncertainty-aware planning enhances information seeking in large language models. *arXiv:2402.03271*.

Ito, Nobuhiro, Yuya Suzuki, and Akiko Aizawa. 2020. From natural language instructions to complex processes: Issues in chaining trigger action rules. *arXiv:2001.02462*.

Jain, Ayush, Pushkal Katara, Nikolaos Gkanatsios, et al. 2024. ODIN: A single model for 2D and 3D perception. *arXiv:2401.02416*. https://doi.org/10.1109/CVPR52733.2024.00342

Jayannavar, Prashant, Anjali Narayan-Chen, and Julia Hockenmaier. 2020. Learning to execute instructions in a Minecraft dialogue. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2589–2602. https://doi.org/10.18653/v1/2020.acl-main.232

Jermsurawong, Jermsak and Nizar Habash. 2015. Predicting the structure of cooking recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 781–786. https://doi.org/10.18653/v1/D15-1090

Jiang, Juyong, Fan Wang, Jiasi Shen, et al. 2024. A survey on large language models for code generation. *arXiv:2406.00515*.

Jiang, Yiwei, Klim Zaporojets, Johannes Deleu, Thomas Demeester, and Chris Develder. 2020. Recipe instruction semantics corpus (RISeC): Resolving semantic structure and zero anaphora in recipes. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 821–826. https://doi.org/10.18653/v1/2020.aacl-main.82

Jiang, Yiwei, Klim Zaporojets, Johannes Deleu, et al. 2022. CookDial: A dataset for task-oriented dialogs grounded in procedural documents. *Applied Intelligence*, 53(4):4748–4766. https://doi.org/10.1007/s10489-022-03692-0

Joshi, Abhinav, Areeb Ahmad, P., et al. 2023. ScriptWorld: Text based environment for learning procedural knowledge. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2023/566

Jung, Yuchul, Jihee Ryu, Kyung-Min Kim, et al. 2010. Automatic construction of a large-scale situation ontology by mining how-to instructions from the Web. *Journal of Web Semantics*, 8(2–3):110–124. https://doi.org/10.1016/j.websem.2010.04.006

Kalithasan, Namasivayam, Arnav Tuli, Vishal Bindal, Himanshu Gaurav Singh, Parag Singla, and Rohan Paul. 2024. Learning to recover from plan execution errors during robot manipulation: A neuro-symbolic approach. *arXiv:2405.18948*. https://doi.org/10.1109/IROS58592.2024.10801831

Kamath, Aishwarya and Rajarshi Das. 2018. A survey on semantic parsing. *ArXiv*, abs/1812.00978.

Kazeminejad, Ghazaleh and Martha Palmer. 2023. Event semantic knowledge in procedural text understanding. In *Proceedings of the 12th Joint Conference on*

*Lexical and Computational Semantics (*SEM 2023)*, pages 388–398. https://doi.org/10.18653/v1/2023.starsem-1.33

Khan, Md. Al Masrur, Md Rashed Jaowad Khan, Abul Tooshil, et al. 2020. A systematic review on reinforcement learning-based robotics within the last decade. *IEEE Access*, 8:176598–176623. https://doi.org/10.1109/ACCESS.2020.3027152

Kiddon, Chloé, Ganesa Th Ponnuraj, avam, et al. 2015. *Mise en Place:* Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 982–992. https://doi.org/10.18653/v1/D15-1114

Kiddon, Chloé, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, pages 329–339. https://doi.org/10.18653/v1/D16-1032

Kim, Geunwoo, Pierre Baldi, and Stephen Marcus McAleer. 2023. Language models can solve computer tasks. *arXiv:2303.17491*.

Kim, Najoung and Sebastian Schuster. 2023. Entity tracking in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, pages 3835–3855. https://doi.org/10.18653/v1/2023.acl-long.213

Ko, Po Chen, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B. Tenenbaum. 2023. Learning to act from actionless videos through dense correspondences. *arXiv:2310.08576*.

Kocbek, Mateja, Gregor Jost, Marjan Hericko, et al. 2015. Business process model and notation: The current state of affairs. *Computer Science and Information Systems*, 12(2):509–539. https://doi.org/10.2298/CSIS140610006K

Koupaee, Mahnaz and William Yang Wang. 2018. WikiHow: A large scale text summarization dataset. *arXiv:1810.09305*.

Kourani, Humam, Alessandro Berti, et al. 2024. Process modeling with large language models. *arXiv:2403.07541*. https://doi.org/10.1007/978-3-031-61007-3_18

Krishna, Ranjay, Kenji Hata, Frederic Ren, et al. 2017. Dense-captioning events in videos. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 706–715. https://doi.org/10.1109/ICCV.2017.83

Kumaran, Vikram, Dan Carpenter, Jonathan Rowe, et al. 2024. Procedural level generation in educational games from natural language instruction. *IEEE Transactions on Games*. https://doi.org/10.1109/TG.2024.3392670

Kwon, Taewan and Sangyong Lee. 2025. OrderSum: Semantic sentence ordering for extractive summarization. *CoRR*, abs/2502.16180.

Lachmy, Royi, Valentina Pyatkin, Avshalom Manevich, et al. 2022. Draw me a flower: Processing and grounding abstraction in natural language. *Transactions of the Association for Computational Linguistics*, 10:1341–1356. https://doi.org/10.1162/tacl_a_00522

Ladhak, Faisal, Esin Durmus, Claire Cardie, et al. 2020. WikiLingua: A new benchmark dataset for cross-lingual abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4034–4048. https://doi.org/10.18653/v1/2020.findings-emnlp.360

Lai, Hanyu, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. AutoWebGLM: A large language model-based Web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5295–5306. https://doi.org/10.1145/3637528.3671620

Lal, Yash Kumar, Vanya Cohen, Nathanael Chambers, et al. 2024. CaT-Bench: Benchmarking language model understanding of causal and temporal dependencies in plans. *arXiv preprint arXiv:2406.15823*. https://doi.org/10.18653/v1/2024.emnlp-main.1077

Le, Thang and Luu Anh Tuan. 2024. Extractive summarization with text generator. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 157–174. https://doi.org/10.18653/v1/2024.naacl-long.9

Li, Belinda Z., Maxwell Nye, and Jacob Andreas. 2021. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*,

pages 1813–1827. https://doi.org/10.18653/v1/2021.acl-long.143

Li, Kailin, Lixin Yang, Zenan Lin, et al. 2024. FAVOR: Full-body AR-driven virtual object rearrangement guided by instruction text. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014*, pages 3136–3144. https://doi.org/10.1609/aaai.v38i4.28097

Li, Kenneth, Aspen K. Hopkins, David Bau, et al. 2022a. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv:2210.13382*.

Li, Mingchen and Lifu Huang. 2023. Understand the dynamic world: An end-to-end knowledge informed framework for open domain entity state tracking. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, pages 842–851. https://doi.org/10.1145/3539618.3591781

Li, Qian, Jianxin Li, Jiawei Sheng, et al. 2022b. A survey on deep learning event extraction: Approaches and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 35(5):6301–6321. https://doi.org/10.1109/TNNLS.2022.3213168

Li, Xinghang, Di Guo, Huaping Liu, et al. 2022c. REVE-CE: Remote embodied visual referring expression in continuous environment. *IEEE Robotics and Automation Letters*, 7(2):1494–1501. https://doi.org/10.1109/LRA.2022.3141150

Li, Xinye, Zunwen Zheng, Qian Zhang, et al. 2025. ScEdit: Script-based assessment of knowledge editing. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2032–2052. https://doi.org/10.18653/v1/2025.findings-acl.104

Li, Yang, Jiacong He, Xin Zhou, et al. 2020. Mapping natural language instructions to mobile UI action sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pages 8198–8210. https://doi.org/10.18653/v1/2020.acl-main.729

Li, Yujia, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022d. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097. https://doi.org/10.1126/science.abq1158, PubMed: 36480631

Li, Zhuang, Lizhen Qu, and Gholamreza Haffari. 2020. Context dependent semantic parsing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2509–2521. https://doi.org/10.18653/v1/2020.coling-main.226

Liang, Zhenwen, Ye Liu, Tong Niu, Xiangliang Zhang, Yingbo Zhou, and Semih Yavuz. 2024. Improving LLM reasoning through scaling inference computation with collaborative verification. *arXiv:2410.05318*.

Lin, Angela, Sudha Rao, Asli Celikyilmaz, et al. 2020. A recipe for creating multimodal aligned datasets for sequential tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4871–4884. https://doi.org/10.18653/v1/2020.acl-main.440

Lin, Jinzhou, Han Gao, Rongtao Xu, et al. 2023. Advances in embodied navigation using large language models: A survey. *arXiv:2311.00530*.

Lin, Yu-An, Chen-Tao Lee, Chih-Han Yang, Guan-Ting Liu, and Shao-Hua Sun. 2024. Hierarchical programmatic option framework. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*.

Liu, Evan Zheran, Kelvin Guu, Panupong Pasupat, et al. 2018. Reinforcement learning on Web interfaces using workflow-guided exploration. In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*. OpenReview.net.

Liu, Xiao, Yansong Feng, Jizhi Tang, et al. 2022. Counterfactual recipe generation: Exploring compositional generalization in a realistic scenario. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7354–7370. https://doi.org/10.18653/v1/2022.emnlp-main.497

Long, Reginald, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Volume 1: Long Papers*, pages 1456–1465. https://doi.org/10.18653/v1/P16-1138

Lu, Yujie, Pan Lu, Zhiyu Chen, et al. 2023. Multimodal procedural planning via dual

text-image prompting. *arXiv:2305.01795*. https://doi.org/10.18653/v1/2024.findings-emnlp.641

Lyu, Qing, Li Zhang, and Chris Callison-Burch. 2021. Goal-oriented script construction. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 184–200. https://doi.org/10.18653/v1/2021.inlg-1.19

MacMahon, Matt, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*, pages 1475–1482.

Maeta, Hirokuni, Tetsuro Sasada, and Shinsuke Mori. 2015. A framework for procedural text understanding. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 50–60. https://doi.org/10.18653/v1/W15-2206

Makoviychuk, Viktor, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac gym: High performance GPU-based physics simulation for robot learning. *arXiv:2108.10470*.

Marc-Alexandre, Côté, Ákos Kádár, et al. 2018. TextWorld: A learning environment for text-based games. In *Computer Games - 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Revised Selected Papers*, volume 1017 of *Communications in Computer and Information Science*, pages 41–75. https://doi.org/10.1007/978-3-030-24337-1_3

Marin, Javier, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2019. Recipe1M+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Matuszek, Cynthia, Evan V. Herbst, Luke Zettlemoyer, et al. 2012. Learning to parse natural language commands to a robot control system. In *International Symposium on Experimental Robotics*. https://doi.org/10.1007/978-3-319-00065-7_28

Mavroudi, Effrosyni, Triantafyllos Afouras, and Lorenzo Torresani. 2023. Learning to ground instructional articles in videos through narrations. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15155–15167. https://doi.org/10.1109/ICCV51070.2023.01395

Mazumder, Sahisnu and Oriana Riva. 2021. FLIN: A flexible natural language interface for Web navigation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, pages 2777–2788. https://doi.org/10.18653/v1/2021.naacl-main.222

Mees, Oier, Lukás Hermann, Erick Rosete-Beas, et al. 2021. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7:7327–7334. https://doi.org/10.1109/LRA.2022.3180108

Miech, Antoine, Dimitri Zhukov, Jean-Baptiste Alayrac, et al. 2019. HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips. In *ICCV*. https://doi.org/10.1109/ICCV.2019.00272

Miglani, Shivam and Neil Yorke-Smith. 2020. NLtoPDDL: One-shot learning of PDDL models from natural language process manuals. In *ICAPS'20 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS'20)*.

Misra, Dipendra, Jaeyong Sung, Kevin Lee, et al. 2014. Tell me Dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 35:281–300. https://doi.org/10.1177/0278364915602060

Misra, Dipendra Kumar, Andrew Bennett, Valts Blukis, et al. 2018. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2667–2678. https://doi.org/10.18653/v1/D18-1287

Misra, Dipendra Kumar, Kejia Tao, Percy Liang, et al. 2015. Environment-driven lexicon induction for high-level instructions. 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)*, pages 992–1002. https://doi.org/10.3115/v1/P15-1096

Modi, Ashutosh, Tatjana Anikina, Simon Ostermann, et al. 2016. InScript: Narrative texts annotated with script information. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Moghe, Nikita, Patrick Xia, Jacob Andreas, et al. 2024. Interpreting user requests in the context of natural language standing instructions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4043–4060. https://doi.org/10.18653/v1/2024.findings-naacl.255

Munkhdalai, Tsendsuren, Manaal Faruqui, and Siddharth Gopal. 2024. Leave no context behind: Efficient infinite context Transformers with infini-attention. *arXiv:2404.07143*.

Mysore, Sheshera, Zachary Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. 2019. The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 56–64. https://doi.org/10.18653/v1/W19-4007

Nabizadeh, Nima, Dorothea Kolossa, and Martin Heckmann. 2020. MyFixit: An annotated dataset, annotation tool, and baseline methods for information extraction from repair manuals. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2120–2128.

Nair, Suraj, Eric Mitchell, Kevin Chen, et al. 2021. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. *arXiv:2109.01115*.

Nakano, Reiichiro, Jacob Hilton, Suchir Balaji, et al. 2021. WebGPT: Browser-assisted question-answering with human feedback. *arXiv:2112.09332*.

Nandy, Abhilash, Manav Kapadnis, et al. 2023. CLMSM: A multi-task learning framework for pre-training on procedural text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8793–8806. https://doi.org/10.18653/v1/2023.findings-emnlp.589

Nandy, Abhilash, Yash Kulkarni, et al. 2024. Order-based pre-training strategies for procedural text understanding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 821–828. https://doi.org/10.18653/v1/2024.naacl-short.74

Narayan-Chen, Anjali, Prashant Jayannavar, and Julia Hockenmaier. 2019. Collaborative dialogue in minecraft. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, pages 5405–5415. https://doi.org/10.18653/v1/P19-1537

Naveed, Humza, Asad Ullah Khan, Shi Qiu, et al. 2024. A comprehensive overview of large language models. *arXiv:2307.06435*.

Nayyar, Rashmeet Kaur and Siddharth Srivastava. 2025. Autonomous option invention for continual hierarchical reinforcement learning and planning. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*. https://doi.org/10.1609/aaai.v39i18.34163

Nguyen, Dai Quoc, Dat Quoc Nguyen, Cuong Xuan Chu, et al. 2017. Sequence to sequence learning for event prediction. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Volume 2: Short Papers*, pages 37–42.

Nishimura, Taichi, Atsushi Hashimoto, Yoshitaka Ushiku, et al. 2021. Structure-aware procedural text generation from an image sequence. *IEEE Access*, 9:2125–2141. https://doi.org/10.1109/ACCESS.2020.3043452

Ostermann, Simon, Ashutosh Modi, Michael Roth, et al. 2018. MCScript: A novel dataset for assessing machine comprehension using script knowledge. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Ostermann, Simon, Michael Roth, and Manfred Pinkal. 2019. MCScript2.0: A machine comprehension corpus focused on script events and participants. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 103–117. https://doi.org/10.18653/v1/S19-1012

Ou, Jiefu, Arda Uzunoglu, Benjamin Van Durme, et al. 2024. WorldAPIs: The world is worth how many APIs? A thought experiment. *arXiv:2407.07778*. https://doi.org/10.1609/aaai.v39i23.34683

Ouyang, Long, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35.

Padmakumar, Aishwarya, Jesse Thomason, Ayush Shrivastava, et al. 2022. TEACh: Task-driven embodied agents that chat. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event*, pages 2017–2025. https://doi.org/10.1609/aaai.v36i2.20097

Page, Matthew J., Joanne E. McKenzie, Patrick M. Bossuyt, et al. 2021a. The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *International Journal of Surgery*, 88:105906. https://doi.org/10.1016/j.ijsu.2021.105906, PubMed: 33789826

Page, Matthew J., David Moher, Patrick M. Bossuyt, et al. 2021b. PRISMA 2020 explanation and elaboration: Updated guidance and exemplars for reporting systematic reviews. *BMJ*, 372. https://doi.org/10.1136/bmj.n160, PubMed: 33781993

Pan, Liang Ming, Jingjing Chen, Jianlong Wu, et al. 2020. Multi-modal cooking workflow construction for food recipes. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1132–1141. https://doi.org/10.1145/3394171.3413765

Park, Hogun and Hamid Reza Motahari-Nezhad. 2018. Learning procedures from text: Codifying how-to procedures in deep neural networks. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018*, pages 351–358. https://doi.org/10.1145/3184558.3186347

Patil, Shishir G., Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*.

Payan, Justin, Swaroop Mishra, Mukul Singh, et al. 2023. Instructexcel: A benchmark for natural language instruction in excel. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4026–4043. https://doi.org/10.18653/v1/2023.findings-emnlp.265

Puig, Xavier, Kevin Ra, Marko Boben, et al. 2018. Virtualhome: Simulating household activities via programs. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pages 8494–8502. https://doi.org/10.1109/CVPR.2018.00886

Qi, Y., Q. Wu, P. Anderson, et al. 2020. Reverie: Remote embodied visual referring expression in real indoor environments. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9979–9988. https://doi.org/10.1109/CVPR42600.2020.01000

Qian, Chen, Lijie Wen, Akhil Kumar, et al. 2020. An approach for process model extraction by multi-grained text classification. In *Advanced Information Systems Engineering*, pages 268–282. https://doi.org/10.1007/978-3-030-49435-3_17

Qin, Yujia, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*.

Rafailov, Rafael, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23.

Raghu, Dinesh, Shantanu Agarwal, Sachindra Joshi, et al. 2021. End-to-end learning of flowchart grounded task-oriented dialogs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 4348–4366. https://doi.org/10.18653/v1/2021.emnlp-main.357

Rajvanshi, Abhinav, Karan Sikka, Xiao Lin, et al. 2023. SayNav: Grounding large language models for dynamic planning to navigation in new environments. In *International Conference on Automated Planning and Scheduling*. https://doi.org/10.1609/icaps.v34i1.31506

Ramakrishnan, Ramya, Ethan Elenberg, Hashan Narangodage, et al. 2023.

Multi-step dialogue workflow action prediction. *arXiv preprint arXiv:2311.09593*.
Regneri, Michaela, Alex Koller, and Manfred Pinkal. 2010. Learning script knowledge with Web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988.
Rein, David, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling*.
Ri, Ryokan, Yufang Hou, Radu Marinescu, et al. 2022. Finding sub-task structure with natural language instruction. In *Proceedings of the First Workshop on Learning with Natural Language Supervision*, pages 1–9. https://doi.org/10.18653/v1/2022.lnls-1.1
Rim, Kyeongmin, Jingxuan Tu, Bingyang Ye, et al. 2023. The coreference under transformation labeling dataset: Entity tracking in procedural texts using event models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12448–12460. https://doi.org/10.18653/v1/2023.findings-acl.788
Rodkin, Ivan, Yuri Kuratov, Aydar Bulatov, and Mikhail Burtsev. 2024. Associative recurrent memory transformer. *arXiv:2407.04841*.
Rojowiec, Robin, Jana Götze, Philipp Sadler, et al. 2020. From "before" to "after": Generating natural language instructions from image pairs in a simple visual domain. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 316–326. https://doi.org/10.18653/v1/2020.inlg-1.38
Roth, Michael and Talita Anthonio. 2021. UnImplicit shared task report: Detecting clarification requirements in instructional text. In *Proceedings of the 1st Workshop on Understanding Implicit and Underspecified Language*, pages 28–32. https://doi.org/10.18653/v1/2021.unimplicit-1.4
Rula, Anisa and Jennifer D'Souza. 2023. Procedural text mining with large language models. In *Proceedings of the 12th Knowledge Capture Conference 2023*, pages 9–16. https://doi.org/10.1145/3587259.3627572
Sakaguchi, Keisuke, Chandra Bhagavatula, et al. 2021. proScript: Partially ordered scripts generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2138–2149. https://doi.org/10.18653/v1/2021.findings-emnlp.184
Salvador, Amaia, Nicholas Hynes, Yusuf Aytar, et al. 2017. Learning cross-modal embeddings for cooking recipes and food images. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3068–3076. https://doi.org/10.1109/CVPR.2017.327
Scalise, Rosario, Shen Li, Henny Admoni, et al. 2018. Natural language instructions for human–robot collaborative manipulation. *The International Journal of Robotics Research*, 37:558–565. https://doi.org/10.1177/0278364918760992
Schank, Roger C. and Robert P. Abelson. 2013. *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*. Psychology Press. https://doi.org/10.4324/9780203781036
Schick, Timo, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.
Scialom, Thomas, Sylvain Lamprier, Benjamin Piwowarski, et al. 2019. Answers unite! Unsupervised metrics for reinforced summarization models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3246–3256. https://doi.org/10.18653/v1/D19-1320
Sen, Procheta, Xi Wang, Ruiqing Xu, et al. 2023. Task2KB: A public task-oriented knowledge base. https://doi.org/10.1609/aaai.v37i13.27086
Shao, Yutong and Ndapa Nakashole. 2020. ChartDialogs: Plotting from natural language instructions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3559–3574. https://doi.org/10.18653/v1/2020.acl-main.328
Shen, Bokui, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, et al. 2021. iGibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots*

and Systems (IROS), pages 7520–7527. https://doi.org/10.1109/IROS51168.2021.9636667

Shi, Haochen, Zhiyuan Sun, Xingdi Yuan, et al. 2024. OPEx: A component-wise analysis of LLM-centric agents in embodied instruction following. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 622–636. https://doi.org/10.18653/v1/2024.acl-long.37

Shi, Qi, Qian Liu, Bei Chen, et al. 2022. LEMON: Language-based environment manipulation via execution-guided pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 471–485. https://doi.org/10.18653/v1/2022.findings-emnlp.33

Shi, Tianlin, Andrej Karpathy, Linxi Fan, et al. 2017. World of bits: An open-domain platform for Web-based agents. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3135–3144.

Shridhar, Mohit, Jesse Thomason, Daniel Gordon, et al. 2020a. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/CVPR42600.2020.01075

Shridhar, Mohit, Xingdi Yuan, Marc-Alex, et al. 2020b. ALFWorld: Aligning text and embodied environments for interactive learning. *arXiv:2010.03768*.

Singh, Himanshu Gaurav, Vishal Bindal, Arnav Tuli, et al. 2022. Learning neuro-symbolic programs for language-guided robotic manipulation. *arXiv:2211.06652*.

Snell, Charlie, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv:2408.03314*.

Song, Chan Hee, Jiaman Wu, Clayton Washington, et al. 2023. LLM-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2998–3009. https://doi.org/10.1109/ICCV51070.2023.00280

Srinet, Kavya, Yacine Jernite, Jonathan Gray, et al. 2020. CraftAssist instruction parsing: Semantic parsing for a voxel-world assistant. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4693–4714. https://doi.org/10.18653/v1/2020.acl-main.427

Steinert, Maurício and Felipe Rech Meneguzzi. 2020. Planning domain generation from natural language step-by-step instructions. In *2020 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS@ ICAPS)*.

Storks, Shane, Qiaozi Gao, Yichi Zhang, et al. 2021. Tiered reasoning for intuitive physics: Toward verifiable commonsense language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4902–4918. https://doi.org/10.18653/v1/2021.findings-emnlp.422

Strathearn, Carl and Dimitra Gkatzia. 2022. Task2Dial: A novel task and dataset for commonsense-enhanced task-based dialogue grounded in documents. In *Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*, pages 187–196. https://doi.org/10.18653/v1/2022.dialdoc-1.21

Sun, Chenkai, Tie Xu, ChengXiang Zhai, et al. 2023. Incorporating task-specific concept knowledge into script learning. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3026–3040. https://doi.org/10.18653/v1/2023.eacl-main.220

Sun, Liangtai, Xingyu Chen, Lu Chen, et al. 2022. META-GUI: Towards multi-modal conversational agents on mobile GUI. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 6699–6712. https://doi.org/10.18653/v1/2022.emnlp-main.449

Tandon, Niket, Bhavana Dalvi, et al. 2019. WIQA: A dataset for "what if..." reasoning over procedural text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6076–6085. https://doi.org/10.18653/v1/D19-1629

Tandon, Niket, Keisuk Sakaguchi, Bhavana Dalvi, et al. 2020. A dataset for tracking entities in open domain procedural text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language*

Processing (EMNLP), pages 6408–6417. https://doi.org/10.18653/v1/2020.emnlp-main.520

Tang, Chen, Ben Abbatematteo, Jiaheng Hu, et al. 2024. Deep reinforcement learning for robotics: A survey of real-world successes. *arXiv:2408.03539*.

Tellex, Stefanie, Thomas Kollar, Steven Dickerson, et al. 2011a. Approaching the symbol grounding problem with probabilistic graphical models. *AI Magazine*, 32:64–76. https://doi.org/10.1609/aimag.v32i4.2384

Tellex, Stefanie, Thomas Kollar, Steven Dickerson, et al. 2011b. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011*, pages 1507–1514. https://doi.org/10.1609/aaai.v25i1.7979

Toshniwal, Shubham, Sam Wiseman, Karen Livescu, et al. 2021. Chess as a testbed for language model state tracking. In *AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v36i10.21390

Toyama, Daniel, Philippe Hamel, Anita Gergely, et al. 2021. AndroidEnv: A reinforcement learning platform for Android. *arXiv:2105.13231*.

Uzunoglu, Arda, Gözde Şahin, and Abdulfattah Safa. 2024. PARADISE: Evaluating implicit planning skills of language models with procedural warnings and tips dataset. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 10085–10102. https://doi.org/10.18653/v1/2024.findings-acl.599

Uzunoglu, Arda and Gözde Gül Sahin. 2023. Benchmarking procedural language understanding for low-resource languages: A case study on Turkish. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, IJCNLP 2023 -Volume 1: Long Papers*, pages 804–819. https://doi.org/10.18653/v1/2023.ijcnlp-main.52

Vogel, Adam and Daniel Jurafsky. 2010. Learning to follow navigational directions. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 806–814.

Wang, Hao, Guosheng Lin, Steven C. H. Hoi, et al. 2022. Learning structural representations for recipe generation and food retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3363–3377.

Wang, Haoyu, Hongming Zhang, Yueguan Wang, et al. 2023. Are all steps equally important? Benchmarking essentiality detection in event processes. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4048–4056. https://doi.org/10.18653/v1/2023.emnlp-main.246

Wang, Lei, Chen Ma, Xueyang Feng, et al. 2024a. A survey on large language model based autonomous agents. *Frontiers in Computer Science*, 18(6):186345. https://doi.org/10.1007/s11704-024-40231-1

Wang, Xingyao, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024b. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, JMLR.org.

Wanzare, Lilian, Aless Zarcone, Stefan Thater, and Manfred Pinkal. 2017. Inducing script structure from crowdsourced event descriptions via semi-supervised clustering. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 1–11. https://doi.org/10.18653/v1/W17-0901

Wanzare, Lilian D. A., Aless, ra Zarcone, et al. 2016. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Weston, Jason, Antoine Bordes, Sumit Chopra, et al. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings*.

Wikipedia. 2024. Business process model and notation. https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation

Wu, Gu, e, Jing Qian, et al. 2024. ARTIST: Automated text simplification for task guidance in augmented reality. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3613904.3642772

Wu, Te Lin, Alex Spangher, Pegah Alipoormolabashi, Marjorie Freedman, Ralph Weischedel, and Nanyun Peng. 2022. Understanding multimodal procedural knowledge by sequencing

multimodal instructional manuals. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4525–4542. https://doi.org/10.18653/v1/2022.acl-long.310

Wu, Te-Lin, Caiqi Zhang, Qingyuan Hu, et al. 2023a. Learning action conditions from instructional manuals for instruction understanding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, pages 3023–3043. https://doi.org/10.18653/v1/2023.acl-long.170

Wu, Xueqing, Sha Li, and Heng Ji. 2023. OpenPI-C: A better benchmark and stronger baseline for open-vocabulary state tracking. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7213–7222. https://doi.org/10.1515/chem-2023-0099

Wu, Yue, Xuan Tang, Tom M. Mitchell, et al. 2023b. SmartPlay: A benchmark for LLMs as intelligent agents. *arXiv:2310.01557*.

Xiang, Wei and Bang Wang. 2019. A survey of event extraction from text. *IEEE Access*, 7:173111–173137. https://doi.org/10.1109/ACCESS.2019.2956831

Xu, Frank F., Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, et al. 2025. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks. *arXiv:2412.14161*.

Xu, Nancy, Sam Masling, Michael Du, et al. 2021. Grounding open-domain instructions to automate Web support tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, pages 1022–1032. https://doi.org/10.18653/v1/2021.naacl-main.80

Yagcioglu, Semih, Aykut Erdem, Erkut Erdem, et al. 2018. RecipeQA: A challenge dataset for multimodal comprehension of cooking recipes. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1358–1368. https://doi.org/10.18653/v1/D18-1166

Yang, John, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. 2023. InterCode: Standardizing and benchmarking interactive coding with execution feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.

Yang, Yue, Artemis Panagopoulou, Qing Lyu, et al. 2021. Visual goal-step inference using wikiHow. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2167–2179. https://doi.org/10.18653/v1/2021.emnlp-main.165

Yao, Shunyu, Howard Chen, John Yang, et al. 2022. WebShop: Towards scalable real-world Web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*.

Yao, Shunyu, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Ye, Seonghyeon, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. 2024. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*.

Ye, Xi, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. 2025. LongProc: Benchmarking long-context language models on long procedural generation. *arXiv preprint arXiv:2501.05414*.

Yuan, Siyu, Jiangjie Chen, Ziquan Fu, et al. 2023. Distilling script knowledge from large language models for constrained language planning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4303–4325. https://doi.org/10.18653/v1/2023.acl-long.236

Zan, Daoguang, Bei Chen, Fengji Zhang, et al. 2023. Large language models meet NL2Code: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, pages 7443–7464. https://doi.org/10.18653/v1/2023.acl-long.411

Zang, Xiaoxue, Ashwini Pokle, Marynel Vázquez, et al. 2018. Translating navigation instructions in natural language to a high-level plan for behavioral robot navigation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2657–2666. https://doi.org/10.18653/v1/D18-1286

Zellers, Rowan, Ari Holtzman, Yonatan Bisk, et al. 2019. HellaSwag: Can a machine

really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800. https://doi.org/10.18653/v1/P19-1472

Zeng, Qingtian, Xiaojie Tang, Weijian Ni, et al. 2020. Missing procedural texts repairing based on process model and activity description templates. *IEEE Access*, 8:12999–13010. https://doi.org/10.1109/ACCESS.2020.2965160

Zhang, Danyang, Hongshen Xu, Zihan Zhao, et al. 2023a. Mobile-Env: An evaluation platform and benchmark for LLM-GUI interaction. *arXiv:2305.08144*.

Zhang, Danyang, Zhennan Shen, Rui Xie, et al. 2024. Mobile-Env: Building Qualified Evaluation Benchmarks for LLM-GUI Interaction. *arXiv:2305.08144*.

Zhang, Hongming, Muhao Chen, Haoyu Wang, et al. 2020. Analogous process structure induction for sub-event sequence prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1541–1550. https://doi.org/10.18653/v1/2020.emnlp-main.119

Zhang, Jingwei, Xin Wu, and Yi Cai. 2023. CLEVR-Implicit: A diagnostic dataset for implicit reasoning in referring expression comprehension. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12820–12830. https://doi.org/10.18653/v1/2023.emnlp-main.791

Zhang, Li. 2022. Reasoning about procedures with natural language processing: A tutorial. *arXiv:2205.07455*.

Zhang, Li, Qing Lyu, and Chris Callison-Burch. 2020. Reasoning about goals, steps, and temporal ordering with WikiHow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 4630–4639. https://doi.org/10.18653/v1/2020.emnlp-main.374

Zhang, Li, Hainiu Xu, Abhinav Kommula, et al. 2024. OpenPI2.0: An improved dataset for entity tracking in texts. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 166–178. https://doi.org/10.18653/v1/2024.eacl-long.10

Zhang, Li, Hainiu Xu, Yue Yang, et al. 2023b. Causal reasoning of entities and events in procedural texts. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 415–431. https://doi.org/10.18653/v1/2023.findings-eacl.31

Zhang, Shengyu, Linfeng Dong, Xiaoya Li, et al. 2023c. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Zhang, Tianyi, Li Zhang, Zhaoyi Hou, et al. 2024a. PROC2PDDL: Open-domain planning representations from texts. In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2024)*, pages 13–24. https://doi.org/10.18653/v1/2024.nlrse-1.2

Zhang, Yi, Sujay Kumar Jauhar, Julia Kiseleva, et al. 2021a. Learning to decompose and organize complex tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2726–2735. https://doi.org/10.18653/v1/2021.naacl-main.217

Zhang, Yixin, Yoko Yamakata, and Keishi Tajima. 2021. Supplementing omitted named entities in cooking procedural text with attached images. In *2021 IEEE 4th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 199–205. https://doi.org/10.1109/MIPR51284.2021.00037

Zhang, Yixin, Yoko Yamakata, and Keishi Tajima. 2022. MIAIS: A multimedia recipe dataset with ingredient annotation at each instructional step. In *Proceedings of the 1st International Workshop on Multimedia for Cooking, Eating, and Related APPlications*, CEA++ '22, pages 49–52. https://doi.org/10.1145/3552485.3554938

Zhang, Yizhe, Jiarui Lu, and Navdeep Jaitly. 2024. Probing the multi-turn planning capabilities of LLMs via 20 question games. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1495–1516. https://doi.org/10.18653/v1/2024.acl-long.82

Zhang, Zhihan, Xiubo Geng, Tao Qin, et al. 2021b. Knowledge-aware procedural text understanding with multi-stage training. In *WWW '21: The Web Conference 2021, Virtual Event*, pages 3512–3523. https://doi.org/10.1145/3442381.3450126

Zhao, Yingxiu, Bowen Yu, Bowen Li, et al. 2023. Causal document-grounded dialogue pre-training. In *EMNLP:2023:main*, pages 7160–7174.

https://doi.org/10.18653/v1/2023.emnlp-main.443

Zhou, Luowei, Chenliang Xu, and Jason J. Corso. 2018. Towards automatic learning of procedures from Web instructional videos. In *AAAI Conference on Artificial Intelligence*, pages 7590–7598. https://doi.org/10.1609/aaai.v32i1.12342

Zhou, Shuyan, Frank F. Xu, Hao Zhu, et al. 2023. WebArena: A realistic Web environment for building autonomous agents. *arXiv:2307.13854*.

Zhou, Shuyan, Pengcheng Yin, and Graham Neubig. 2022. Hierarchical control of situated agents through natural language. In *Proceedings of the Workshop on Structured and Unstructured Knowledge Integration (SUKI)*, pages 67–84. https://doi.org/10.18653/v1/2022.suki-1.8

Zhou, Shuyan, Li Zhang, Yue Yang, et al. 2022. Show me more details: Discovering hierarchies of procedures from semi-structured Web data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2998–3012. https://doi.org/10.18653/v1/2022.acl-long.214

Zhou, Yilun, Julie Shah, and Steven Schockaert. 2019. Learning household task knowledge from WikiHow descriptions. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 50–56.

Zhu, Yuqi, Shuofei Qiao, Yixin Ou, et al. 2024. KnowAgent: Knowledge-augmented planning for LLM-based agents. *arXiv preprint arXiv:2403.03101*. https://doi.org/10.18653/v1/2025.findings-naacl.205