Grounding with Structure: Exploring Design Variations of Grounded Human-AI Collaboration in a Natural Language Interface

HYO JIN DO, IBM Research, USA
MICHELLE BRACHMAN, IBM Research, USA
CASEY DUGAN, IBM Research, USA
JAMES M. JOHNSON, IBM Research, USA
JULIA LAUER, IBM, USA
PRIYANSHU RAI, IBM Research, India
QIAN PAN, IBM Research, USA

Selecting an effective utterance among countless possibilities that match a user's intention poses a challenge when using natural language interfaces. To address the challenge, we leveraged the principle of least collaborative effort in communication grounding theory and designed three grounded conversational interactions: 1) a *grounding interface* allows users to start with a provisional input and then invite a conversational agent to complete their input, 2) a *multiple grounding interface* presents multiple inputs for the user to select from, and 3) a *structured grounding interface* guides users to write inputs in a structure best understood by the system. We compared our three grounding interfaces to an ungrounded control interface in a crowdsourced study (N=80) using a natural language system that generates small programs. We found that the grounding interfaces reduced cognitive load and improved task performance. The structured grounding interface further reduced speaker change costs and improved technology acceptance, without sacrificing the perception of control. We discuss the implications of designing grounded conversational interactions in natural language systems.

CCS Concepts: • **Human-centered computing → HCI design and evaluation methods**; **User studies**; **Empirical studies in HCI**; **Natural language interfaces**.

Additional Key Words and Phrases: Natural Language Interface; Conversational User Interface; Conversational Grounding

ACM Reference Format:
Hyo Jin Do, Michelle Brachman, Casey Dugan, James M. Johnson, Julia Lauer, Priyanshu Rai, and Qian Pan. 2024. Grounding with Structure: Exploring Design Variations of Grounded Human-AI Collaboration in a Natural Language Interface. *Proc. ACM Hum.-Comput. Interact.* 8, CSCW2, Article 363 (November 2024), 27 pages. https://doi.org/10.1145/3686902

1 Introduction

Natural language (NL) interfaces are increasingly provided to support novices in using technical systems more easily. For example, NL interfaces are used to create short programs [59, 94], configure mashups of IoT services [50], construct diagrams [102], or request information from a database [1].

Authors' Contact Information: Hyo Jin Do, hjdo@ibm.com, IBM Research, Cambridge, MA, USA; Michelle Brachman, michelle.brachman@ibm.com, IBM Research, Cambridge, MA, USA; Casey Dugan, cadugan@us.ibm.com, IBM Research, Cambridge, MA, USA; James M. Johnson, jmjohnson@us.ibm.com, IBM Research, Cambridge, MA, USA; Julia Lauer, julia.lauer@ibm.com, IBM, New York, NY, USA; Priyanshu Rai, priyanshu.rai@ibm.com, IBM Research, Pune, India; Qian Pan, qian.pan@ibm.com, IBM Research, Cambridge, MA, USA.

ACM 2573-0142/2024/11-ART363
https://doi.org/10.1145/3686902

The recent advancements in NL systems using large language models (LLMs) have made it more accessible than ever before to accomplish engineering tasks like programming and data manipulations using naturalistic prompts [2, 29, 47]. Our work focuses on the use of an NL interface for a workflow automation system that integrates business applications or data. For example, a user can write a sentence such as "when there is a new attendee on Eventbrite, add a subscriber on MailChimp" and the system will generate a short program that automates the workflow. The use of NL interfaces is promising because people find it natural, intuitive, and easy to use such interfaces [88].

However, NL systems face the challenge of *abstraction matching*, which refers to the difficulty of formulating an utterance with an appropriate level of abstraction that matches the system's capabilities, such as selecting suitable words, grammar, and structure that the system can take [39, 68, 86]. This challenge becomes more prominent in the tasks involving transactional utterances, where the extraction and recognition of specific linguistic features are particularly essential, as opposed to non-transactional utterances that are less structurally analyzable. Even with the recent emergence of LLMs, discovering an input prompt to yield a desired output is still necessary because the models tend to favor certain prompt formats and paraphrases, i.e. prompt engineering [70, 78, 83]. A prior study found that end-users do prompt engineering opportunistically rather than systematically and struggled to make progress in LLM outputs [103]. Continuous failures to provide a 'matching' input can lead to user frustration, reduce user acceptance of the technology, and make users abandon the system [104].

To address this challenge, Liu et al. proposed *grounded abstraction matching* [68] – when a user submits an NL utterance, the system performs an action and subsequently translates it back to a naturalistic utterance, allowing the user to learn from it. Through interviews, the researchers discovered that this approach promoted explanation and debugging, enhanced users' trust and confidence, and impacted users' mental models. Our work builds on this prior work [68] by designing and comparing various interfaces that implement grounded abstraction matching in a conversational interface. Beyond the benefits identified in the prior work, we further evaluated communication costs and task performance associated with different designs of grounded interactions, which are crucial to consider in order to foster positive human-AI interactions.

To design interfaces to support grounded abstraction matching [68], we employed the principle of least collaborative effort in communication grounding [16]. This principle explains that it takes more effort to design a proper utterance than to design an improper utterance and invite a partner's help. Based on this principle, we designed a conversational *grounding interface* with key design goals: 1) *provisional first input:* the agent first asks the user to compose part of the final sentence rather than a complete sentence, 2) *shared responsibility of input presentations:* the user and the agent take turns to present how they want to write the input, and 3) *collaborative revision reference:* they refer to each other's inputs and revise them. However, these principles may lead to higher speaker costs due to more exchanges with the agent or higher formulation costs due to the need to modify the agent's inputs. We designed a *multiple grounding interface*, which aims to reduce the cost of switching conversational turns by presenting more than one input in each agent's turn. We also designed a *structured grounding interface*, which aims to reduce the cost of formulating a full, unconstrained sentence by allowing users to write keywords or short phrases in a fill-in-the-blank style. We compared these interfaces with an *ungrounded control interface*, which incorporates traditionally established practices for effectively writing an input such as providing best input examples and output explanations.

We conducted a between-subjects experiment with participants from Amazon Mechanical Turk (N=80). Each participant used one of the four interfaces: grounding, multiple grounding, structured grounding, and control interfaces. Participants were given two tasks, with the goal to write a sentence describing a trigger-action program to automate a workflow. They responded to survey questions about their cognitive load, acceptance and perceptions of the system, and experiences with the system. We also measured task performance and communication costs such as speaker change costs and formulation costs.

Our results showed a positive impact of the grounding interfaces, especially when offering guidance for the structure. Overall, the grounding interfaces reduced participants' cognitive load and increased the task performance compared to the control interface. Furthermore, participants using the structured grounding interface succeeded in their task and accepted the system more at a lower speaker change cost and cognitive load than participants using other interfaces. Contrary to our expectations, participants using the structured grounding interface also felt less constrained and more in control than participants using other grounding interfaces.

Our contributions are three-fold. First, we propose a novel conversational interaction design that embodies the principle of least collaborative effort. Second, we compare various designs of grounding interfaces in terms of communication cost and task performance and suggest that grounding interactions with structured input fields are the most effective design without constraining users. Lastly, we offer design implications for designing conversational grounding interfaces in NL systems, providing insights and recommendations for the future development of human-AI collaboration.

In the following sections, we review prior literature in Section 2, explain our formative study in Section 3, and outline our design goals in Section 4. Sections 5 and 6 detail the methodology and results of our experiment, followed by the discussion and the conclusion in Sections 7 and 8.

2 Background and Related Work

We begin by discussing the abstraction matching problem, which is a critical challenge in natural language systems, and review the relevant research and technology designs that address this issue.

2.1 Abstraction Matching Problem

Our work focuses on a goal-oriented NL system that enables users to automate a workflow by integrating data and business applications. NL interfaces have been increasingly developed for this task automation purpose [30, 36, 82] and other similar programming tasks [59, 101], database query processing [1], IoT and mashup configurations [67], data visualizations [88], and web page designs [51]. NL interfaces are intuitive, natural, and easier to learn compared to traditional interfaces that require users to use programming languages and software protocols to operate the system [32, 97]. The recent emergence of large language models (LLMs) supports the promising future for their utilization in various use cases even more [2, 29, 47, 68]. Graphical user interfaces (GUIs) are an alternative method that has been traditionally used for task automation including commercial applications (e.g., IBM App Connect[1], Microsoft Power Automate[2], Zapier[3], IFTTT[4]). Huang et al. compared a crowd-sourced NL interface and a GUI when creating trigger-action program rules and found that they are comparable in terms of outcome quality [35]. However, GUIs often have a steep learning curve as it depends heavily on visual cues (e.g. buttons, icons) and designing effective GUIs is also challenging. For example, GUIs in the task automation context may expose either too many or few functionalities [36] which makes it difficult for users to discover ways to build their programs [10]. It is also challenging to automate complex programs because

[1]https://www.ibm.com/cloud/app-connect
[2]https://powerautomate.microsoft.com/digital-process-automation/
[3]https://zapier.com
[4]https://ifttt.com/

they may require handcrafted composition of entities by users, whereas NL interfaces can reduce such complex GUI interactions [10, 56].

One caveat of NL interfaces is that users have difficulty selecting an NL input from the near-infinite space of naturalistic inputs that they believe the system will reliably map to a satisfactory solution, which is referred to as *abstraction matching* [68, 86]. This problem also exists in NL systems that utilize LLMs, which require users to figure out effective inputs to elicit better responses, known as prompt engineering [69]. Specifically, LLMs tend to favor certain prompt formats, such as open-ended question prompts (e.g., 'Where did John invite Mark?'), which outperform restrictive prompts (e.g., 'John invited Mark to home. Output True or False') [3]. Sarkar et al. articulated the problem as "fuzzy", because the systems can now afford an extremely wider range of input abstractions than traditional models, and consequently, users find it more difficult than before to learn and generalize from the results on how to write effective inputs [86].

The abstraction matching problem is not new and has existed in traditional NL systems including task-oriented chatbots. For example, researchers found that people use a surprisingly large variety of words to refer to the same object or commands in terminals and dialogue systems, as known as the "vocabulary problem" [24], which led to research efforts to create shared vocabulary of concepts and relations [95]. Intent detection and action mapping are related concepts but they focus on different aspects. Abstraction matching focuses on the user's responsibility in which a user tries to adapt their utterances in a way that is appropriate for the system to understand. Intent detection focuses on the system's responsibility of identifying the user's intention given an utterance, while action mapping determines the system's specific executions. Goal-oriented NL systems incorporate various deterministic modules that are triggered based on detected intents in the input sentence. These models often cannot parse and understand arbitrary NL inputs reliably for goal-oriented interactions [48, 79]. Therefore, users have to learn how to write inputs that the system can reliably process and find it difficult to form accurate mental models about system capabilities [66, 71]. People also struggle to evaluate whether their NL inputs are improving in a way that is best for the system to operate [38].

2.2 Natural Language Interface Designs

Traditional NL systems have included various nudging and training methods [96] to address the abstraction matching problem. The first method is breaking down a larger interaction required to perform a complex task into smaller interactions to support user input. Researchers have enabled users to do this in a variety of ways such as through creating multiple steps to incrementally perform the task [87], dividing the task into multiple sub-tasks [100], and most commonly, using conversational interfaces. Researchers have increasingly advocated the use of conversational interfaces for complex tasks including end-user programming [13, 59, 94], data science [22], and mashup configurations [50]. A conversational interface can effectively provide structural guidance and ask clarification questions through multiple conversational turns [13], but creates more opportunities for errors as the complexity of conversations increases [22]. Researchers have explored ways to improve the basic conversational interface. For example, the direction of communication (i.e., who is leading the conversation) changed users' perceptions of the system in a word game [4].

Another way to help users write reliable NL input is through input suggestions. Systems often provide generic examples of commands that are available to manage user expectations of the expected linguistic structure and in/out-of-scope operations [84]. Even recent technologies such as Apple's Siri[5], Amazon's Alexa[6], and ChatGPT[7] present prompt examples. A study of multiple methods of providing examples showed that in-situ command suggestions were best at supporting the discovery of commands [90]. Similarly, NL systems also provide auto-completion to predict next words [9, 39, 88, 93, 102] and adaptive command templates [25, 73]. Adding additional web forms in addition to NL commands, such as drop-down menus, multiple choice selection, and sliders, may help communicate the system understanding to the user and allow the user to clarify any ambiguities or errors [34, 46, 57, 88]. For example, Analyza [20] and Datatone [26] provide widgets either for generalization or for disambiguation. While these web forms might also work in a task with limited capabilities, these forms will be not well-suited for NL interfaces with expanded functionalities, including iterations and contextual information. Explanations of the results could also help users rephrase their inputs in order to receive a better outcome in future interactions. Systems provide explanations in a variety of ways, like messages and suggestions [61, 62], paraphrased and annotated user inputs [30], visual explanations [18, 49], and providing the context of a conversational interaction [37].

NL systems can guide or even force users to write reliable input through constrained NL. For example, CONVO is a constrained NL-based programming system in which commands have to be stated exactly for the system to understand [94]. However, there are mixed results regarding the benefits and drawbacks of constrained and unconstrained NL. When using constrained NL for a language learning game, users performed similarly to those using unconstrained language but had lower mental effort [75]. An artificial language also outperformed NL in accuracy and user preference for human-robot interaction (HRI) [76]. However, constrained NL may work less successfully as interactions get more complex [8] and reduce controllability, thereby decreasing user acceptance of the system [21]. Another study looked at four levels of restriction in NL along the 'Formality Continuum' for semantic web search, from keyword-based to NL sentences, and found that unconstrained NL was ranked best by users [45]. Following these best practices, we present interface designs that allow users to write a provisional input that a conversational agent then builds upon, suggest reliable inputs and explanations in-situ grounded on the user's previous inputs and the system results, and provide structures that contrain users to write their NL input.

2.3 Grounding in Human-AI Communication

Liu et al. proposed the concept of *grounded abstraction matching* [68] in the context of end-user programming in which the NL system translates the generated code back into an NL utterance. This allows the user to observe this NL utterance and develop their mental model for future interactions with the system. Through an interview study comparing grounded and ungrounded techniques, the researchers demonstrated mostly qualitative evidence that the grounded approach helps debug errors, increases users' trust and confidence in using the system, and shapes user mental models of what can and can't be done [68]. Building on this study, our research broadens the understanding of the grounded approach, particularly on *how to design grounded abstraction matching*.

Communication grounding is the process in which the participants of a conversation attempt to update the common ground [11, 15, 16]. While formulating a comprehensible utterance at first attempt would be ideal, repair is often inevitable if it fails and grounding can become critical in this process. There are two phases in communication: the presentation phase, in which a speaker presents an utterance, and the acceptance phase, in which an addressee provides evidence of whether they understood the utterance. The theory includes the principle of least collaborative

[5]https://www.apple.com/siri/
[6]https://alexa.amazon.com
[7]https://chat.openai.com/

effort in which participants in a conversation try to minimize their collaborative effort – the work they both do from the initiation of each contribution to its mutual acceptance. The principle writes that "speakers often realize that it will take a more collaborative effort to design a proper utterance than to design an improper utterance and enlist their addressees' help." We apply the principle of least collaborative effort to the design of an NL interface by allowing a user to collaborate with AI to design a proper utterance. To effectively test this grounding interface design, we selected challenging tasks where a user is unlikely to succeed at initial utterance and tries to repair with the least collaborative effort. We investigated the following research question with the associated hypothesis: **RQ1. How did the grounding interface impact the *cognitive load*?**

H1. Cognitive load of participants will decrease in the grounding interface compared to the ungrounding interface because they share the responsibility with the system to formulate the utterance that generates a correct output.

Designing for least collaborative effort could result in costs. We focus on two of these costs in this work, speaker change costs and formulation costs. Speaker change costs refer to the effort it takes to switch between speakers, which could be reduced by doing more within a turn. Formulation costs are a result of planning and authoring the NL in a conversation. The theory explains that "it costs more to formulate perfect than imperfect utterances [16]." We designed two additional NL grounding interfaces that focus on reducing these costs. The Multiple grounding interface suggests multiple NL utterances at each turn to reduce speaker change costs. The Structured grounding interface provides a template for the input structure to reduce formulation costs. We therefore explored the following research question with corresponding hypotheses: **RQ2. How did the grounding interfaces impact the *communication costs*?**

H2-1. The Multiple grounding interface will reduce speaker change costs than the other interfaces because the agent presents more information in each turn.

H2-2. The Structured grounding interface will reduce formulation costs than the other interfaces because the interface allows a user to write keyword-style inputs than a free-form input.

While grounding with least collaborative effort is a natural process in human-human communication, researchers have begun to see its opportunities to support human-AI interactions, especially agent-human communication. Some argue that achieving 'common ground' is critical in designing usable conversational interactions with systems [11, 53, 65]. Researchers identified opportunities for improving dialog interactions through communication grounding, in particular during repair and use of evidence [33]. Cho et al. studied grounding in conversations between a user and a personal assistant, recommending that the user and the agent should be conversational partners, rather than bystanders [15]. Further studies using communication grounding theory found that users benefit from agent suggestions [5], users prefer to have an understanding of the state of agent knowledge [58], and users have a potential preference for explicit feedback over implicit feedback [105]. However, people may fundamentally question grounding with agents [17] and even be averse to having a proactive agent attempting to collaborate with the user [63]. Building on existing AI research leveraging communication grounding, our study aims to enhance our empirical understanding of the impact of various grounding interface designs on user perceptions of the system. Hence, our next research question was: **RQ3. How did the grounding interfaces impact user *perceptions,* i.e. technology acceptance, perceptions of control, and perceptions of constraint of the NL system?**

H3-1. Technology acceptance will increase when using grounding interfaces compared to the ungrounding interface because users find the system more helpful and improve their productivity by completing their provisional input and providing suggestions.

H3-2. The Structured grounding interface will reduce the perceptions of control and increase the perceptions of constraint than other interfaces because users are not able to freely modify the main structure of the input.

Further, researchers found benefits of grounding in achieving a successful outcome. For example, Gergie, et al. demonstrated that shared visual information can reduce collaborative effort by helping users recognize when a worker is performing incorrect actions, thereby achieving faster and better task performance [27]. While many prior works focus on grounding through visual technology, we posed the following research question for NL systems: **RQ4. How did the grounding interfaces impact *task performance*?**

H4. The task performance will increase when using grounding interfaces compared to the ungrounding interface because participants can better understand the system capabilities through grounding and able to formulate better utterances.

There have been existing research and technology designs that used the grounding theory and its principles [5, 55, 91, 92]. For example, Ashktorab et al. used the grounding principles to design repair strategies in a conversational breakdown and investigated user preferences [5]. Unlike existing research, our study is unique in the embodiment of least collaborative effort principle in the design of a conversational NL interface with the purpose of bridging the gap between user mental models and the system capabilities. We isolated individual elements of interest (e.g., number of utterances, input template) across three grounding interface designs and conducted a systematic and empirical evaluation of their effectiveness across multiple dimensions.

3 Formative Study

The goal of our formative study was to understand how users would communicate in a conversational way to direct an agent to generate an automation flow in the context of our system. Thus we ran a wizard-of-oz formative study, in which a user conversed with a researcher acting as an intelligent agent to generate short programs in real time.

3.1 Method

Each session involved one participant and two researchers, a moderator and a researcher simulating a conversational agent. The moderator created a unique task for each session so that the conversational agent simulator would not know the participant's goal. The user's task was to generate a small trigger-action program using NL (e.g., Fig. 5). The participant conversed with the simulated agent through Slack to generate the flow in NL. The simulated agent provided two types of NL feedback to the user: 1) rephrasing the user's input to confirm understanding, and 2) asking clarification questions with multiple options (e.g. "What would you like to do with Gmail? For example, you can send an email or create a contact."). The agent also provided screenshots of the generated programs based on the user's inputs. Finally, the moderator asked follow-up questions about what was helpful and challenging. The sessions took place over video conferencing software and were recorded, with the participants' permission. The study sessions were up to 30 minutes, with the task portion lasting up to 15 minutes. We recruited 11 participants from internal messaging channels within a large IT company. Three researchers transcribed participants' responses, grouped and labeled related responses independently in an affinity diagram, and discussed common responses. Next, we report three relevant themes of responses that guided our interface designs in Section 4.

3.2 Preliminary Findings

3.2.1 Missing keywords in the first interaction. Only four participants (36%) provided the important keywords critical for success in their initial interaction with the intelligent agent. A few participants mentioned that providing a structured template might be useful to address this problem.

> *The agent could even like say hey, I have some similar scenario flow do you want to reuse those instead of create a new one, just make small changes that will be more productive, because then I don't have to create from scratch.* -F1

3.2.2 Structuring input. Participants were allowed to choose how to structure their interaction with the intelligent agent facilitator. We found that there were two main strategies participants used to formulate their utterances. Eight participants (73%) broke the utterance down into pieces and provided them to the intelligent agent one by one. The other three participants (27%) attempted to provide all or most of the information about the program in one utterance to the intelligent agent.

> *Ideally, I would have liked to just write it in one sentence and be able to send that message, but then I liked that when it was wrong, we could like, look at parts of it when the agent said like 'is this correct', and I was like 'no', and then I knew which parts we were looking at I was like 'okay, oh, I didn't say Salesforce'.* -F5

3.2.3 Agent feedback and suggestions. Six participants appreciated that the agent rephrased and confirmed the flow.

> *It was very helpful when she asked me 'does this look right?' because it allows me to have some kind of guidance of what I'm doing and to make the small corrections* -F1

For those who broke up their inputs, the intermediate feedback allowed them to check their program to gain more confidence in what they were doing, get guidance on what they could do better in the future, and make small changes based on what the intelligent agent generated so far.

> *It was helpful when I was using the wrong words or the wrong type of action, she told me the options available* -F6

Overall, participants benefited from the intermediate feedback and suggestions they received from the simulated conversational agent, and liked to break up their NL input into parts, but needed more guidance knowing the key elements and structure of the input. Along with these results, we used the principle of least communication effort in communication grounding theory to design interactions aimed at reducing the effort of abstraction matching in the following study.

4 Interface Design and System

We discuss our design goals and the four conversational interfaces that we implemented for the between-subject design experiment.

4.1 Design Goals

Inspired by the formative study and the principle of least collaborative effort in communication grounding theory [16], we developed three design goals for grounding interfaces for NL systems.

(1) **Provisional first interaction:** The grounding interfaces do not expect a user to make a perfect NL input in their first attempt. The interfaces instead ask a user to write a provisional input at first, like which applications to connect. In contrast, the ungrounded NL interface expects a user to make a proper input in their first attempt.
(2) **Shared responsibility of input presentations**: In the grounding interfaces, it is a shared responsibility between the agent and the user to create an input that meets the user's goal and

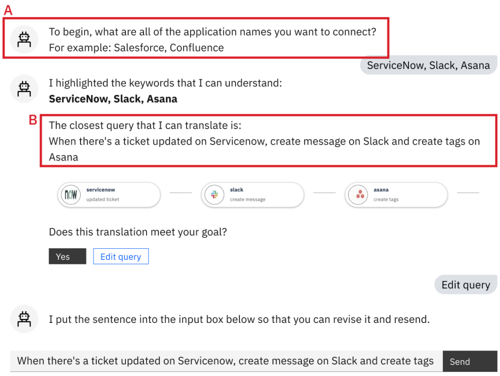

Fig. 1. Grounding interface. The agent invites a user to write a provisional input at first (A). The agent generates a workflow and translates the flow back to an NL utterance (B), which allows the user to observe and revise in subsequent interactions.

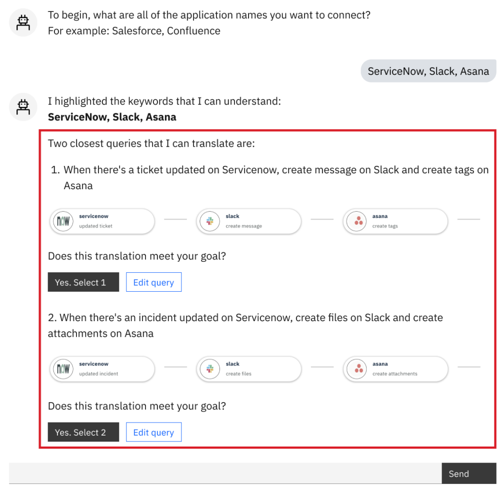

Fig. 2. Multiple grounding interface. As highlighted, the agent presents more than one input to reduce speaker change costs.

system capabilities, thereby the user's effort is less than the total effort. In the ungrounded interface, it is solely the user's responsibility to formulate the input.

(3) **Collaborative revision reference**: In the grounding interfaces, the agent and the user build on each other's inputs and revise collaboratively. In the ungrounded interface, the user revises their own previous inputs.

In addition to the key design goals of grounding, we followed three general design goals in all interfaces. First, explainable AI (XAI) research has supported the use of explanations to justify algorithmic decisions or suggestions [80]. Therefore, we highlighted keywords in the user input to help users understand what words are parsed in the system. Second, visual feedback is generally more effective to support user decisions than text-only feedback [23]. Thus, agent responses provided flow diagrams to represent each agent input. Third, all interfaces provided an example input before a user interacts with the system for the first time, which is common in NL interfaces.

4.2 Conversational Interface Designs

Based on the design goals, we implemented four conversational interfaces: grounding, multiple grounding, structured grounding, and control interfaces. Considering the iterative nature of their grounding approach, it was appropriate to leverage the conversational interface. As illustrated in Fig. 1, agent inputs are prefaced with a bot icon on the left, user inputs are given a dark gray background. The input field is at the bottom of each interface. For simplicity of the study, users and the agent alternated turns, each providing one input or response at a time. We iterated on the design by collecting feedback from independent pilot testers to validate the understandability and usability of our system.

4.2.1 Grounding interface. In the grounding interface (Fig. 1), the agent begins by asking the user to provide the application names they would like to connect, based on our provisional first interaction design goal. Next, the agent generates the workflow diagram and translates the diagram

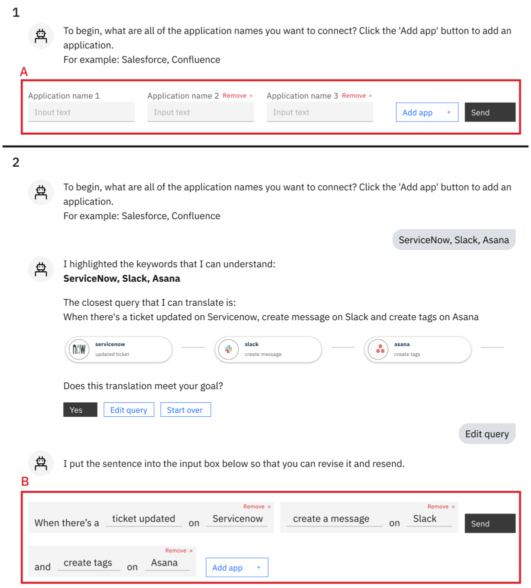

Fig. 3. Structured grounding interface. The user is given specific input boxes to provide their provisional input (A). In the following interactions (2), a user writes keywords to fill in the blanks of the structured input boxes (B) rather than writing a sentence in free form, to reduce formulation costs.

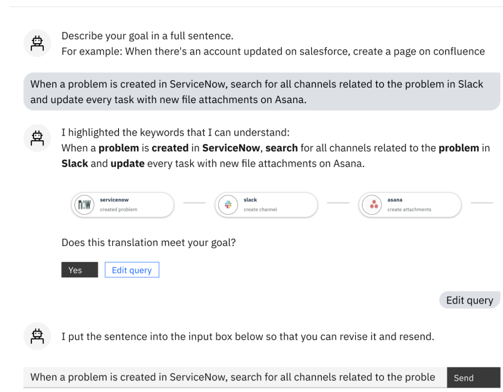

Fig. 4. Ungrounded control interface. Similar to other interfaces, the design was guided by the established best practices to guide abstraction matching such as suggesting input examples and providing visual feedback with highlighted keywords as explanations.

back to an NL utterance, based on our shared responsibility of input presentations design goal. We describe how the agent's input is generated in Section 4.3. The user can modify the agent's input by clicking 'Edit query' button, based on our collaborative revision reference design goal, or type their own input into the input box.

4.2.2 Multiple grounding interface. The multiple grounding interface (Fig. 2) aims to reduce speaker change costs by having the agent provide multiple options for NL utterances. The multiple grounding interface includes all of the features of the grounding interface with one difference: when the agent responds to the user's input after the first interaction, the agent provides two suggested flows rather than just one. Ideally, providing two agent input options will lead to a higher likelihood that one is correct, thus reducing the number of turns needed by the user and agent to converge on a correct input. Researchers empirically tested different number of flows and found that our system generally predicted the correct flow within the first two flows, implying that providing more than two flows is unnecessary.

4.2.3 Structured grounding interface. The structured grounding interface (Fig. 3) aims to reduce the formulation cost of modifying the agent's input. Formulation cost is the effort associated with deciding what words and structure should be used to communicate [16]. The structured grounding interface includes the features from the grounding interface with one difference: the agent provides a structured template for the user to write an input, rather than a free-form input box. For the user's first interaction with the conversational agent, they can see two input boxes to enter application names and can add or remove input boxes (Fig. 3-1). Once the agent suggests an input, the user can edit it in a template format (Fig. 3-2). The interface forces users to write an input in the format 'When there's a [trigger event] on [application], [action] on [application]', which they can add or remove action phrases (i.e., 'and [action] on [application]'). Pilot testers mentioned that sometimes it takes too much effort to revise the agent input due to the constrained nature of this particular interface and it would be easier to start over. Based on their suggestion, we implemented a 'start over' button that allows users to restart the conversation by asking for application names again and 10 participants used this feature as a result.

4.2.4 Control interface. This is the control interface for comparison that does not include the grounding design goals (Fig. 4). Inspired by the ungrounded interface proposed in the prior work [68], we strengthened the interface design by incorporating best practices in conventional NL systems to avoid the straw man fallacy, which is oversimplifying the opponent's argument. The agent begins by providing the user with an example input and asks the user to write an NL sentence. The user can type in a full sentence to try to generate their flow correctly. Then, the agent responds with the generated flow diagram, highlights recognized words in the user's input as explanations, and enables users to easily modify their own previous input by automatically copying the previous input into the input field. In case of ambiguous words, the system unhighlighted them to imply that the system did not recognize those words. The system showed error messages when the system detects a pre-defined error type, which includes Empty input (a user submits an empty input), Time-out (the system is not able to process the input within a reasonable time), Missing app name (no application name is detected), and Network error (network connection is unstable). These error messages were infrequent, yet they effectively mitigated user frustration when errors happen. All of these strategies were also included in the other interfaces so that we can isolate the effect of grounding and its variations.

4.3 Natural Language System Architecture for Workflow Automation

We designed our NL interfaces in the context of workflow automation system, which generates short programs that integrate applications. We focus on trigger-action programming capabilities that are commonly used in similar automation tools (e.g., Zapier[8], IFTTT[9]) or IoT programming contexts. When a user writes a trigger-action program in an NL sentence, the system identifies applications, a trigger event, actions, and objects. Then, it creates a short program that is represented by a flow diagram. Fig. 5 illustrates an example flow diagram generated by an input 'when there's a new incident on ServiceNow, send a message on Slack, and send an email on Gmail'. The system identifies ServiceNow, Slack, and Gmail as applications, a new incident as a trigger event, send as an action, and a message and an email as objects.

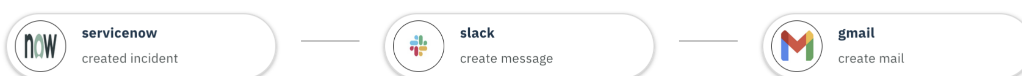

Fig. 5. An example program represented by a flow diagram. An NL input "when there's a new incident on ServiceNow, send a message on Slack, and send an email on Gmail' will generate the program.

The system has four main steps that take place to process an NL input, generate a program, and write an NL utterance for the agent to suggest: 1) parsing, 2) matching, 3) planning, and 4) generating inputs. The system first parses the input, using an Abstract Meaning Representation [6, 7, 77]. It

[8]https://zapier.com/
[9]https://ifttt.com/

Table 1. Demographic profiles of participants. The total number of participants for each factor is reported in the Total column. The number of participants who were associated with each condition is reported. The mean and standard deviation are reported for factors measured using a Likert scale.

Factors	Range	Total	Control	Grounding	Multiple	Structured
Gender	Male	44	12	15	9	8
	Female	36	8	5	11	12
Age	18-29 years	17	4	4	3	6
	30-39 years	29	5	6	12	6
	40-49 years	24	5	8	4	7
	50-59 years	5	3	0	1	1
	60 years or older	5	3	2	0	0
Education	Less than high school degree	1	1	0	0	0
	High school degree or equivalent	16	2	3	6	5
	Some college but no degree	16	6	2	1	7
	Associates degree	5	2	1	1	1
	Bachelors degree	37	7	12	12	6
	Graduate degree	5	2	2	0	1
AI experience	I have heard about AI in the news, friends, or family	56	9	14	18	15
	I closely follow AI-related news	15	4	5	2	4
	I have some work experience and/or formal education related to AI	9	7	1	0	1
Writing skills	1: Poor - 7: Excellent	5.71 (1.02)	5.85 (0.81)	5.90 (0.97)	5.65 (1.35)	5.45 (0.88)
Interaction with chatbots	1: Never - 7: Very Frequently	4.43 (1.84)	4.95 (1.93)	4.5 (1.96)	3.80 (1.40)	4.45 (1.96)
Programming experience	1: No experience - 10: Expert	2.26 (1.06)	2.50 (1.00)	2.50 (1.05)	2.00 (0.97)	2.05 (1.19)

uses this parsing information to determine important keywords in the input such as application names, objects, actions, as well as the structure of the input (e.g., which parts of the input refer to the trigger and actions). This parsing works best if the utterance uses NL, rather than any type of pseudocode, and a structure that indicates the trigger and subsequent actions, like "when [ ], do [ ] and then [ ]." The system then attempts to match those keywords to a knowledge graph. The knowledge graph is a static structure that contains the meta-data for the possible triggers and actions, along with descriptions. From the knowledge graph, the system obtains top matches for trigger events, actions, objects, and applications for the flow. For example, given an application name, the graph can return strongest candidates of action and object by calculating the closest nodes from the application node, therefore handles synonyms or ambiguous words. Using the candidates provided by the knowledge graph, an AI planner [40–44] determines the order of the flow components. The system can generate one or more flows. Proposed flows are produced in a native YAML format, similar to JSON. For the grounding interfaces, the system generates the agent's suggested input by translating the YAML to an NL sentence. More specifically, we created a sentence by inserting application names, actions, objects, and trigger events included in the YAML into a predetermined sentence template. In all grounding interfaces, we used the same sentence template to control its effect such as 'When there is a/an [trigger event] on [application], [action] [object] on [application], and [action] [object] on [application].' We used the same system across conditions to control the performance of the system. Technical details of the system can be found in this paper [10]. While we chose a conventional system design in this experiment, our findings could generalize to recent NL systems using LLMs as the problem and the design of grounded abstraction matching is applicable to any NL systems as we explained in Section 2.

5 Method

We conducted a between-subjects design experiment with four conditions. Each condition was associated with one of the interface designs: grounding, multiple grounding, structured grounding, and control interfaces.

5.1 Participants

We recruited crowdworkers from the Amazon Mechanical Turk platform. People who were 18 years old or older, located in the United States, and native English speakers were eligible to participate in our study. We recruited people who have less than moderate programming experience [89] to focus on the target audience of our NL system generating short programs and to control confounding effects. Workers whose number of approved tasks was greater than 1000 and HIT approval rate was greater than 98% were allowed to work on our task, in order to receive quality responses. We also asked an attention check question in the post-task survey, and manually inspected individual responses to enhance data quality. We therefore excluded data from 3 participants out of the 83 total submissions. As reported in Table. 1, 80 participants completed the study with valid responses, 20 participants for each condition. We found no significant differences in gender, age group, education, AI experience, writing skills, interaction frequency with chatbots, and programming experience across conditions.

5.2 Study Procedure

We conducted an online user study that lasted less than 30 minutes. We first distributed a qualification survey to get a list of qualified workers who were interested in our study. We invited the qualified participants to the actual task page. Participants signed a consent form and completed a pre-task survey that asked about their background information and prior experience with AI and conversational agents. Then, they read instructions about how to use the interface and completed two tasks. They could move on to the next task when they submitted a correct input for the task or the timer has ended (10 minutes per task at maximum). After completing the tasks, they submitted a post-task survey. Participants were compensated $3.63 for completing the user study (based on US federal minimum wage) and $0.30 for the qualification survey. We provided a bonus payment ($0.25) for each correct task.

5.3 Tasks

The goal of the tasks was to type an NL input into a chat in order to generate a small program that connects business applications. We designed two independent tasks [98], each with a hypothetical scenario and a visual goal flow diagram that a user needs to generate through their NL utterances. The task order was counterbalanced within a condition. The tasks are the following:

(1) Imagine that you work on an IT support team. When a problem is created in ServiceNow, you want to automate a series of tasks. You want to 1) search for all channels related to the problem in Slack and 2) update every task with new file attachments on Asana.
(2) Imagine that you gave a talk to potential customers and you told them that they can email you if they want the slides, which are stored in Dropbox. When there is an incoming mail in your Msoffice inbox, you want to automate a series of tasks: 1) share the slides, 2) make an appointment using Msoffice calendar, and 3) update their contact information on Insightly.

We purposely designed the tasks to be challenging for non-programmers and those unfamiliar with the system to avoid a ceiling effect (the majority of people achieve close to the highest possible score, and the variance in an independent variable becomes attenuated). Also, we would expect that the grounding interfaces would be the most necessary in cases where users have difficulty entering a successful NL input at the first trial. We iteratively adjusted the difficulty of the tasks through separate pilot tests, distinct from the formative study mentioned earlier.

5.4 Measures

We used survey responses and system logs to answer our research questions.

5.4.1 Survey responses. We asked questions about cognitive load to answer RQ1 and questions about user acceptance of the system, perceptions of control and constraint to answer RQ3.

- **Cognitive load**: Participants rated their cognitive load while using the system through four questions: their mental demand, performance, effort, and frustration while doing the tasks. Each question used a 20-point scale, adopted from NASA-TLX [31]. We averaged these four scales to calculate the cognitive load measure (α=.87).
- **Technology acceptance**: Participants rated their acceptance of the system using five questions [54], adopted from the technology acceptance model [19], with a seven-point scale (e.g., 'I would use the system if it was available', 'I found the system to be helpful', 'I found the system to be able to improve my productivity'). We averaged these five scales to calculate the technology acceptance measure (α=.96).
- **Perception of control**: Participants rated their agreement with the following question on a seven-point scale: 'I felt that I was in control of the system' [54].
- **Perception of constraint**: Participants rated their agreement with the following question on a seven-point scale: 'I feel constrained to write the sentence in a certain way.'

5.4.2 System logs. Using system logs, we measured speaker change costs and formulation costs to answer RQ2. We measured the number of correct task outcome to answer RQ4.

- **Speaker change costs**: We counted the number of conversational turns in which the user sent an input. The number of turns can be used as a proxy for speaker change costs because the agent and the user take turns one by one and cannot interrupt while the other is forming a response.
- **Formulation costs**: We measured the time in seconds that users took to write their input in each turn to estimate formulation costs.
- **Task performance**: We measured whether a user's input generated the goal flow successfully for each task (0: fail, 1: success).

5.5 Analysis

When analyzing survey responses such as participants' cognitive load and acceptance of the system, we constructed a linear regression model using the condition as a independent variable. We conducted an F-test to calculate the statistical significance. If there was a significant effect with more than two conditions compared, we ran a post-hoc test with the Bonferroni correction to find pairs with a significant difference. When analysing speaker change costs, formulation costs, and task performance, we constructed a linear mixed-effects regression model using the condition as a fixed effect variable and the task ID and the participant ID as random effect variables to account for the variability between the two tasks and the participants. We calculated the statistical significance using likelihood-ratio chi-squared tests [74]. We ran a post-hoc test with the Bonferroni correction when significant results were found.

6 Results

We show the effects of proposed interfaces on the cognitive load (RQ1), communication costs (RQ2), acceptance and perceptions of the system (RQ3), and task performance (RQ4). We also explore users' interactions based on our design goals. The descriptive statistics of all measures are summarized in Table 2. For linear mixed-effects models, we report full model outputs in the Appendix. All charts depict means and standard errors.

Table 2. Descriptive statistics of results for all conditions. We report the means and standard deviations in parentheses.

RQ	Measure	Grounding	Multiple Grounding	Structured Grounding	Control
1	Cognitive load	13.80 (3.96)	14.30 (4.73)	10.01 (7.57)	15.21 (5.01)
2	Speaker change costs	7.60 (3.80)	6.98 (3.03)	4.48 (2.41)	6.50 (2.59)
	Formulation costs	26.04 (12.41)	26.13 (12.34)	36.41 (26.50)	34.52 (26.8)
3	Technology acceptance	2.18 (1.53)	1.89 (1.18)	3.30 (2.22)	2.24 (2.03)
	Perception of control	2.35 (1.57)	2.65 (1.53)	3.75 (2.10)	3.05 (1.88)
	Perception of constraint	6.2 (1.01)	6.3 (1.45)	5 (2.08)	6.15 (1.69)
4	Task performance	0.35 (0.48)	0.35 (0.48)	0.63 (0.49)	0.23 (0.42)

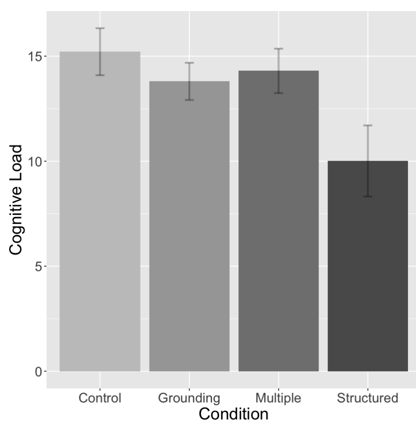

Fig. 6. Impact of interfaces on cognitive load. Participants rated lower cognitive load when using the grounding interfaces, particularly the structured grounding interface, compared to the control interface.

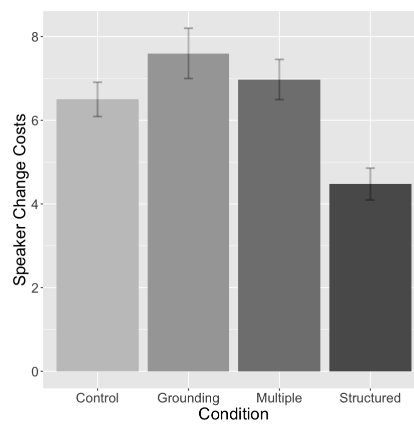

Fig. 7. Impact of interfaces on speaker change costs. The structured grounding interface led to significantly lower speaker change costs than other grounding interfaces.

6.1 Cognitive Load (RQ1)

We found a significant effect of interface on the cognitive load of participants ($F(3, 76) = 3.48, p = .02$). Post-hoc tests revealed that participants reported lower cognitive load when using the structured grounding interface compared to the control interface ($p = .02$) or the multiple grounding interface ($p = .09$). Comparing grounding interfaces (N=60) and the control interface (N=20), the grounding interaction affected cognitive load with a marginal difference ($F(1, 78) = 2.94, p = .09$). Participants reported lower cognitive load when using grounding interfaces (M=12.70, SD=5.86) than those who used the control interface (M=15.21, SD=5.01). We, therefore, support the hypothesis that the grounding interfaces can lower the cognitive load, especially when using the structured grounding interface (H1 accepted).

6.2 Communication Costs (RQ2)

6.2.1 Speaker change costs. We found a significant effect of interface on speaker change costs ($\chi^2(3) = 13.47, p = .004$). Post-hoc tests revealed that the structured grounding interface had significantly lower speaker change costs than the grounding interface ($p = .002$) and the multiple

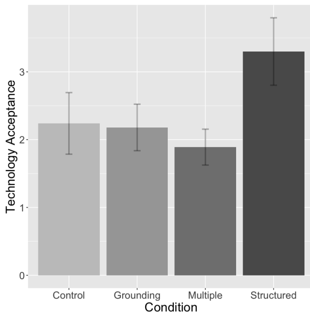

Fig. 8. Impact of interfaces on user acceptance of the system. Participants indicated higher technology acceptance when using the structured grounding interface than the multiple grounding interface.

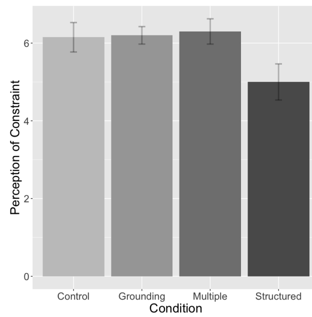

Fig. 9. Impact of interfaces on the perception of constraint. Participants felt less constrained when using the structured grounding interface than the multiple grounding interface.

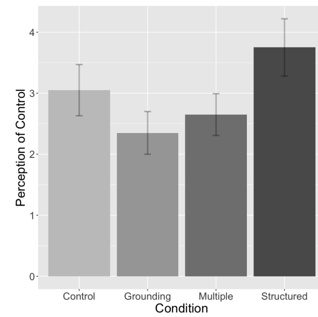

Fig. 10. Impact of interfaces on the perception of control. Participants felt more in control when using the structured grounding interface than the grounding interface.

grounding interface (p = .02). Comparing grounding interfaces (N=60) and the control interface (N=20), there was not a significant impact of grounding on speaker change costs (p = .85). Hence, our hypothesis was not supported that our multiple grounding interface lowered the speaker change costs (H2-1 rejected). Instead, the structured grounding interface (M=4.48, SD=2.41) lowered the speaker change costs compared to the grounding interface (M=7.60, SD=3.80) by 3.12 turns, and compared to the multiple grounding interface (M=6.98, SD=3.03) by 2.5 turns, as reported in Table 2.

To further understand why the multiple grounding interface did not reduce speaker change costs, we considered the Levenshtein distance[10], or the number of character edits to change between the user's input and the agent's first and second input options. Of the 154 cases where users edited agent inputs in the multiple grounding interface, in 128 cases (83%), the user's input was closer to the agent's first input than to the agent's second input. This means that users were making less use of the second input suggested by the agent, reducing the impact of having multiple options.

6.2.2 Formulation costs. We hypothesized that the structured grounding interface would reduce formulation costs by providing an input structure that could reduce the time to modify the agent input. We did not find a significant difference in formulation costs across interfaces (p = .17), thus our hypothesis was not supported (H2-2 rejected). We discuss possible reasons in Section 7.

6.3 Technology Acceptance and Perceptions (RQ3)

We evaluated 1) user acceptance of the system and 2) perceptions of control and constraint. Overall, participants were more likely to accept the structured grounding interface than the multiple grounding interface. Participants felt less constrained and more in control when using the structured grounding interface than other interfaces.

6.3.1 User acceptance of the system. The interface predicted user acceptance of the system with marginal significance ($F(3, 76) = 2.38, p = .08$). Pairwise comparisons revealed that participants accepted the structured grounding interface more than the multiple grounding interface with

[10]https://pypi.org/project/editdistance/

marginal significance ($p = .09$). There was no significant difference in user acceptance of the system between participants using the grounding interfaces (N=60) and the control interface (N=20) ($p = .65$) (H3-1 rejected).

6.3.2 Perceptions of control and constraint. One risk of having the system more involved in the construction of inputs is that users might feel more constrained or less in control of the system. This was a particular concern for the structured grounding interface, which users were not able to freely modify the main structure of their input. We found a significant effect of the interface on the perception of constraint ($F(3, 76) = 2.90, p = .04$) and a marginally significant effect of interface on the perception of control of the system ($F(3, 76) = 2.31, p = .08$). Post-hoc tests revealed that participants who used the structured grounding interface felt marginally less constrained than those who used the multiple grounding interface ($p = .07$) and felt marginally more in control than those in the grounding interface ($p = .09$). Therefore, paradoxically, participants using the structured grounding interface felt less constrained and more in control, rather than the opposite (H3-2 rejected).

6.4 Task Performance (RQ4)

We hypothesized that enabling users to work together with the agent to formulate NL inputs would improve users' ability to generate correct outcomes using the system. Comparing the grounding interfaces (N=60) and the control interface (N=20), the grounding interaction affected the task performance with marginal significance ($\chi^2(1) = 3.83, p = .05$). Participants showed higher task performance when using the grounding interfaces (M=0.44, SD=0.50) compared to the control interface (M=0.23, SD=0.42), thus our hypothesis was supported (H4 supported). When comparing interfaces individually, we found a significant effect of interface on the task performance ($\chi^2(3) = 9.67, p = .02$). Post-hoc tests revealed that the structured grounding interface had a significantly higher task performance than the control interface ($p = .01$). In sum, the grounding interfaces increased the task performance and the structured grounding interface significantly increased the task performance than the control interface. Note that the task performance was evaluated for each task with a binary score 0 (fail) and 1 (success). Therefore, the mean task performance scores in Table 2 imply that successful task completion rate, which are about 35–63% when using the grounding interfaces, while it's only 23% when using the control interface. We successfully designed challenging tasks in order to effectively answer our research questions of whether grounding interfaces could support users when they fail in the first submission, resulting in generally low scores across interfaces.

6.5 Supplementary Analysis: User interactions and Demographic Factors

In this section, we report additional insights on how participants interact with the key design goals of the grounding interface, and the effects of demographic factors. We first measured whether users chose to write application names to start as requested by the agent or write a complete sentence instead (design goal 1). Most participants took advantage of the option to start of with a provisional interaction. In the grounding condition, 16 and 15 participants did so for tasks 1 and 2 respectively, of the 20 participants. In the multiple grounding condition, 14 and 11 participants did so for tasks 1 and 2 respectively. For structured grounding condition, 19 and 17 did so for tasks 1 and 2 respectively.

We measured whether users chose to modify the agent's inputs by clicking 'Edit query' button, rather than writing their own in the textbox, as a proxy to understand whether users shared the responsibility of input formulation (design goal 2 & 3). In the grounding interface, on average 69% ($SD = 24\%$) of user inputs were edits of the agent's input. For the structured grounding, participants edited a previous agent utterance 66% (SD = 12%) of the time. For the multiple grounding interface, 56% (SD = 21%) of inputs were edits of the agent's input. One possible reason for the less usage of agent input in the multiple grounding interface is that providing multiple inputs added extra effort to use the agent inputs, leading users to provide their own inputs instead. To further elaborate *how* users collaborated with the agent, one actual conversation (shortened) from a participant is presented below:

> User: servicenow, slack, asana, asana
> Agent: When there's a ticket updated on Servicenow, create message on Slack and create tags on Asana
> User: When there's a *problem created* on Servicenow, *show slack channels*, *update every task in asana, attach files asana*
> Agent: When there's a problem created on Servicenow, *create channel on Slack* and *create tasks on Asana and create attachments on Asana*
> User: When there's a problem created on Servicenow, create channel on Slack and *updateall* tasks on Asana and create attachments on Asana
> Agent: When there's a problem created on Servicenow, create channel on Slack and updateall tasks on Asana and create attachments on Asana
> User: When there's a problem created on Servicenow, *retrieveall* channel on Slack and updateall tasks on Asana and create attachments on Asana [generates correct flow]

The user first started with a provisional input and the agent returned a complete sentence as a starter. In the rest of the conversation, the user and the agent collaboratively revised the previous inputs by adding, editing, reordering, or correcting keywords (actions, objects) and phrases.

Additionally, we investigated whether there is any interaction effect between any demographic factors and the interface designs. We found significant interaction effects between the age group (encoded numerically) and a few key dependent variables as the following:

(1) Cognitive load ($F(7, 72) = 3.53, p < .01$): Younger participants reported significantly lower cognitive load when using the structured grounding interface, while other interfaces had no significant difference across age groups.
(2) User acceptance of the system ($F(7, 72) = 3.56, p < .01$): Younger participants reported significantly higher acceptance of the system when using multiple and structured grounding interfaces.
(3) Perceptions of control ($F(7, 72) = 2.86, p < .05$): Younger participants reported significantly higher perceptions of control when using multiple and structured grounding interfaces.

There were no other significant interaction effects among demographic factors (e.g., gender, age, education, writing skills, interaction frequency with chatbot). While demographic factors were not our focus, these additional findings imply that the structured grounding interface is even more likely to be accepted by younger participants, reducing cognitive overload and increasing perception of control. The multiple grounding interface also showed similar findings except cognitive load. One possible explanation is their existing familiarity with digital communication, such as texting or messaging apps, positively influenced their experience with our grounding designs. We encourage researchers to further explore the relationship between age and grounding strategies.

7 Discussion

We discuss the implications of designing grounding interfaces, the generalizability to other conversational and NL systems, and the limitations of the study.

7.1 Design Implications

This study advances our empirical knowledge of grounding interface designs to address the abstraction matching problem. Our results further identified that the *structured grounding interface* is the most effective design to implement grounded abstraction matching. Our grounding interfaces embodied the principle of least collaborative effort by enabling users to write a provisional input such as application names at first and inviting a conversational agent to complete the input. Providing system-generated NL utterances and allowing users to easily edit the utterances encouraged users to revise their inputs in a way that matches system capabilities. In fact, we provided empirical evidence that participants edited the previous agent utterance 69% of the times when using the grounding interface. Our results demonstrated that grounding interfaces can reduce users' cognitive load and increase their task performance, while previous research did not yield a significant quantitative difference in cognitive load and task performance between grounded and ungrounded conditions, owing to the variable effects of a think-aloud protocol and easiness of the tasks [68]. The grounding approach also aligns with the objectives of nudging, particularly the 'facilitate' mechanism [12], in which diminishing individuals' mental or physical effort can encourage them to pursue a predefined set of actions. Thus, we recommend designing conversational interfaces that allow users to provide provisional inputs that the system can then help the user to refine.

We also found that the structured grounding interface enhanced the effects of grounding by improving user acceptance of the system without users feeling constrained. The technology acceptance model [19] explains that perceived ease of use is a key factor influencing an individual's acceptance of the system. Participants using the structured grounding interface may have felt it easier to write their input with a given structure, increasing user acceptance of the technology. While NL systems often strive for naturalness by allowing users to create their input in a free form [32, 52, 81, 97], our results suggest that naturalness may not be the most important factor in goal-oriented conversations. This aligns with prior work findings, that moderately constrained languages can yield similar or better acceptance of the system without negatively impacting user experience [75], especially in situations where users do not expect social interaction with the agent [64]. Our work shows that a system can provide a structure for the user's input, leading to improved performance and acceptance, without the user feeling constrained in what they can write. While our study revealed the effects of structuring in the grounding context, it is important to acknowledge that the structuring design may have its own merits in contexts where there is a limited solution space and technical entities to be connected.

Recently, research and applications leveraging large language models have increasingly embraced structured interfaces for effective prompt engineering [60, 99, 100]. For instance, a prompt builder for Midjourney[11] (AI art generator) has been developed to accommodate inputs with specific structures. While an ideal AI model should be able to decipher user intentions from any inputs, adopting a structured interface remains a possible solution considering the limitations of the current technology as well as user challenges associated with prompt engineering. Our research findings provide insights into how these structured interfaces could be further improved by incorporating the grounding approach.

We also found that the structured grounding interface reduced speaker change costs in communication, but did not impact formulation costs. Given that the task performance was the highest in the structured grounding condition, we can infer that participants using the structured grounding interface stopped the conversation early after generating a correct task outcome, thus reducing speaker change costs. We did not find a significant decrease in formulation costs when using the structured interface, possibly due to the additional costs of interacting with UI components (e.g.,

[11] https://promptomania.com/midjourney-prompt-builder/

button, input boxes). Another possible explanation is that the savings in formulation costs may not be significantly correlated with the syntax or structure of the input, but rather the semantic factors such as the use of appropriate keywords and grammar. Therefore, designers should be aware of the trade-off between the costs of system interaction and the costs of grounding, while also identifying the primary source of the formulation costs.

While recent work indicated that multiple replies from a conversational agent could improve user experience [14], we did not find that effect in the multiple grounding condition. One reason could be that the first input generated by the agent was much more effective than the second agent input, so users in the multiple grounding interface condition gained less from knowing the second input compared to users in other conditions who received only one input each time. This aligns with the problem of finding the appropriate 'number of search suggestions' in search engines, which refers to a challenge in information retrieval where it returns too many or too few suggestions when users type in a query. Researchers have devised methods to improve the relevance and diversity of search suggestions, making it more effectively for users to find the information they need [72]. Likewise, we anticipate that the speaker change costs would have been reduced if additional agent inputs had helped users achieve the goal more effectively than receiving just one input each time. One way to generate more relevant and diverse inputs dynamically is to use multiple algorithms, such that each algorithm has been optimized differently to generate inputs (e.g., high precision and high recall algorithms [54]).

7.2 Generalizability of Designs and Results

7.2.1 Applying and generating grounding agent interactions. Our study implemented the ideas from the principle of least collaborative effort using one input (i.e., "what are the apps you want to connect?") to guide users' provisional input and a predetermined template (e.g., "When there's an [event] on [app], [action] on [app]") to provide agent inputs. This allowed us to control the performance and type of inputs that the agent provided for this experiment. In order to apply this type of support in other NL systems, there are two main design decisions to make: 1) what are the questions the system asks the user, and 2) how will the system generate the agent's suggested inputs? We used human intelligence to choose a good question to request provisional input with full knowledge of the tasks. Determining a good question to ask for provisional input in a variety of contexts and tasks is a challenging but beneficial step forward in applying the grounding design studied in the paper and could be a compelling use of AI in assisting. For example, our system could have asked a user to input the first step in their flow, rather than the application names. Future work should explore how to construct these questions to obtain minimal information needed for the system to generate a suggested input. Moreover, agent responses would be generated dynamically, rather than using a static template. To generate the agent's suggested inputs, a system could leverage a set of successful NL inputs from other users or system designers to train a distribution of words using generative NL models, essentially learning one or more templates for successful interactions with the system. Future work could explore whether more variation in the suggested inputs better reduces effort and helps users perform tasks successfully with an NL system.

7.2.2 Grounding beyond conversations. Though we designed and implemented our grounding interfaces in the context of conversational interactions, the ideas could be implemented in other types of NL or constrained NL contexts. For example, many NL systems using basic text input fields [30, 73, 84, 88] or widgets (e.g., drop-down menus, multiple choice selections) [101]. While these interactions lack the traditional 'conversational' visual form, there is still a back-and-forth exchange of information between the user and the system (i.e., a user submits NL input and receive feedback message from the system such as pop-up windows or error texts). Thus, systems with this structure can still allow users to input provisional inputs, provide one or more suggested inputs from the system, and provide structured guidance for the input as in our conversational interface.

7.2.3 Tasks beyond automating workflows. Our tasks to generate flows from NL represent complex problem-solving tasks. We believe that our findings generalize to other tasks in which users must form accurate mental models of AI systems to make reliable requests. For example, a customer service chatbot can adapt the grounding interactions to assist customers more reliably [5]. In cooperative gaming contexts, grounding can be useful to enhance understanding of their AI partner [28]. Enterprise resource planning (ERP) systems can use the grounding approach as they often have restricted NL query interface to retrieve data from knowledge bases [85]. For simple or familiar tasks, grounding interactions may not be necessary because users would already know how to make a reliable input on their first attempt.

7.3 Limitations and Future Work

We intentionally designed difficult tasks as it was reflected in the low average values in task performance. This decision was necessary to determine whether grounding interfaces could support users when systems do not work in the first submission. The decision was also influenced by the previous research [68] in which they suggested that their high success rate could have been responsible for insignificant quantitative differences in cognitive load or usability. Consequently, the task performance among participants was generally low, which might have also lowered the technology acceptance overall. While the task design was appropriate for our research purpose, we anticipate a higher task performance and technology acceptance in real-world applications. We were unable to study our system and interventions over a longer period of time. There might be different impacts of the grounding interfaces over time, where users have a better understanding of the system after using the grounding interface than the control interface, leading to more efficient and successful interactions. We ran our study using Amazon's Mechanical Turk, which has known limitations such as the user population, as in any other crowdsourcing platforms. Therefore, our study is limited to the demographic makeup of the participants. Specifically, the participants consisted of crowdworkers with limited programming experience. While lay users should be able to write a NL input to automate a given workflow using our interface, we do not intend to generalize our results to the population who are familiar with automation tasks or tools. Also, we were only able to consider a limited set of communication costs, while there are other types of communication costs [16]. More research is necessary to explore different types of communication costs in grounding interactions. For example, the delay cost for generating inputs was controlled in this work but should be considered when implementing grounding in future systems where agent inputs are generated by different algorithms. While we only focused on grounding in NL interfaces, we encourage future work to compare other types of interfaces such as GUI interfaces, which still structure the interactions while having less of the grounding conversations.

8 Conclusion

To address the abstraction matching problem, this work advocates for the idea of grounding between a user and an NL system as a way to connect the system's capabilities and the user's intentions. We designed NL interfaces based on the principle of least collaborative effort in communication grounding theory in which a user writes a provisional input at first and invites a conversational agent to collaborate and complete their input. We showed the effectiveness of grounding interactions in reducing the cognitive load of users and increasing task performance. Further, giving more guidance on the input structure in the grounding process reduces the cost of changing speakers and increases acceptance of the system without the user feeling constrained or losing control. Our findings have the potential to benefit the design of a variety of NL systems beyond conversational interfaces and support users in completing complex tasks where users need to develop accurate mental models of the system.

References

[1] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal* 28, 5 (2019), 793–819.

[2] Open AI. 2023. API. https://openai.com/blog/openai-api

[3] Simran Arora, Avanika Narayan, Mayee F Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, and Christopher Re. 2022. Ask me anything: A simple strategy for prompting language models. In *The Eleventh International Conference on Learning Representations*.

[4] Zahra Ashktorab, Casey Dugan, James Johnson, Qian Pan, Wei Zhang, Sadhana Kumaravel, and Murray Campbell. 2021. Effects of communication directionality and AI agent differences in human-AI interaction. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[5] Zahra Ashktorab, Mohit Jain, Q Vera Liao, and Justin D Weisz. 2019. Resilient chatbots: Repair strategy preferences for conversational breakdowns. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–12.

[6] Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based Parsing with Stack-Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 1001–1007.

[7] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*. 178–186.

[8] Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, Roberto Basili, and Daniele Nardi. 2014. Effective and Robust Natural Language Understanding for Human-Robot Interaction.. In *ECAI*. 57–62.

[9] Abraham Bernstein and Esther Kaufmann. 2006. GINO–a guided input natural language ontology editor. In *International semantic web conference*. Springer, 144–157.

[10] Michelle Brachman, Christopher Bygrave, Tathagata Chakraborti, Arunima Chaudhary, Zhining Ding, Casey Dugan, David Gros, Thomas Gschwind, James Johnson, Jim Laredo, et al. 2022. A Goal-Driven Natural Language Interface for Creating Application Integration Workflows. (2022).

[11] Susan E Brennan. 1998. The grounding problem in conversations with and through computers. *Social and cognitive approaches to interpersonal communication* (1998), 201–225.

[12] Ana Caraban, Evangelos Karapanos, Daniel Gonçalves, and Pedro Campos. 2019. 23 ways to nudge: A review of technology-mediated nudging in human-computer interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–15.

[13] Shobhit Chaurasia and Raymond Mooney. 2017. Dialog for language to code. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 175–180.

[14] Eason Chen. 2022. The Effect of Multiple Replies for Natural Language Generation Chatbots. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 1–5.

[15] Janghee Cho and Emilee Rader. 2020. The role of conversational grounding in supporting symbiosis between people and digital assistants. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW1 (2020), 1–28.

[16] Herbert H Clark and Susan E Brennan. 1991. Grounding in communication. (1991).

[17] Leigh Clark, Nadia Pantidi, Orla Cooney, Philip Doyle, Diego Garaialde, Justin Edwards, Brendan Spillane, Emer Gilmartin, Christine Murad, Cosmin Munteanu, Vincent Wade, and Benjamin R. Cowan. 2019. What Makes a Good Conversation? Challenges in Designing Truly Conversational Agents. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300705

[18] Weiwei Cui, Xiaoyu Zhang, Yun Wang, He Huang, Bei Chen, Lei Fang, Haidong Zhang, Jian-Guan Lou, and Dongmei Zhang. 2019. Text-to-viz: Automatic generation of infographics from proportion-related natural language statements. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 906–916.

[19] Fred D Davis. 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly* (1989), 319–340.

[20] Kedar Dhamdhere, Kevin S McCurley, Ralfi Nahmias, Mukund Sundararajan, and Qiqi Yan. 2017. Analyza: Exploring data with conversation. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. 493–504.

[21] Berkeley J Dietvorst, Joseph P Simmons, and Cade Massey. 2018. Overcoming algorithm aversion: People will use imperfect algorithms if they can (even slightly) modify them. *Management Science* 64, 3 (2018), 1155–1170.

[22] Ethan Fast, Binbin Chen, Julia Mendelsohn, Jonathan Bassen, and Michael S Bernstein. 2018. Iris: A conversational agent for complex tasks. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–12.

[23] Michael Fernandes, Logan Walls, Sean Munson, Jessica Hullman, and Matthew Kay. 2018. Uncertainty displays using quantile dotplots or cdfs improve transit decision-making. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[24] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM* 30, 11 (1987), 964–971.

[25] Anushay Furqan, Chelsea Myers, and Jichen Zhu. 2017. Learnability through adaptive discovery tools in voice user interfaces. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 1617–1623.

[26] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G Karahalios. 2015. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th annual acm symposium on user interface software & technology*. 489–500.

[27] Darren Gergle, Robert E Kraut, and Susan R Fussell. 2004. Language efficiency and visual technology: Minimizing collaborative effort with visual information. *Journal of language and social psychology* 23, 4 (2004), 491–517.

[28] Katy Ilonka Gero, Zahra Ashktorab, Casey Dugan, Qian Pan, James Johnson, Werner Geyer, Maria Ruiz, Sarah Miller, David R Millen, Murray Campbell, et al. 2020. Mental models of ai agents in a cooperative game setting. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[29] GitHub. 2023. Copilot. https://github.com/features/copilot

[30] Sumit Gulwani and Mark Marron. 2014. Nlyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 803–814.

[31] Sandra G Hart. 2006. NASA-task load index (NASA-TLX); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 50. Sage publications Sage CA: Los Angeles, CA, 904–908.

[32] Gary G Hendrix. 1982. Natural-language interface. *American Journal of Computational Linguistics* 8, 2 (1982), 56–61.

[33] Wan Ching Ho. 2020. Hello Siri, Why Don't You Understand?–A Study on Grounding of Human and Agent Interlocutors in Dialogue. *ESSLLI & WeSSLLI*, 79.

[34] Enamul Hoque, Vidya Setlur, Melanie Tory, and Isaac Dykeman. 2017. Applying pragmatics principles for interaction with visual analytics. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 309–318.

[35] Ting-Hao'Kenneth' Huang, Amos Azaria, Oscar J Romero, and Jeffrey P Bigham. 2019. InstructableCrowd: Creating IF-THEN Rules for Smartphones via Conversations with the Crowd. *arXiv preprint arXiv:1909.05725* (2019).

[36] Ting-Hao Kenneth Huang, Amos Azaria, and Jeffrey P Bigham. 2016. Instructablecrowd: Creating if-then rules via conversations with the crowd. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 1555–1562.

[37] Mohit Jain, Ramachandra Kota, Pratyush Kumar, and Shwetak N Patel. 2018. Convey: Exploring the use of a context view for chatbots. In *Proceedings of the 2018 chi conference on human factors in computing systems*. 1–6.

[38] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. PromptMaker: Prompt-based Prototyping with Large Language Models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 1–8.

[39] Ellen Jiang, Edwin Toh, Alejandra Molina, Kristen Olson, Claire Kayacik, Aaron Donsbach, Carrie J Cai, and Michael Terry. 2022. Discovering the Syntax and Strategies of Natural Language Programming with Generative Language Models. In *CHI Conference on Human Factors in Computing Systems*. 1–19.

[40] Michael Katz. 2018. Cerberus: Red-black heuristic for planning tasks with conditional effects meets novelty heuristic and enchanced mutex detection. *Ninth International Planning Competition (IPC-9): planner abstracts* (2018), 47–51.

[41] Michael Katz and Shirin Sohrabi. 2020. Reshaping diverse planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34.06. 9892–9899.

[42] Michael Katz, Shirin Sohrabi, Horst Samulowitz, and Silvan Sievers. 2018. Delfi: Online planner selection for cost-optimal planning. *IPC-9 planner abstracts* (2018), 57–64.

[43] Michael Katz, Shirin Sohrabi, and Octavian Udrea. 2020. Bounding Quality in Diverse Planning. *HSDIP 2020* (2020), 49.

[44] Michael Katz, Shirin Sohrabi, Octavian Udrea, and Dominik Winterer. 2018. A novel iterative approach to top-k planning. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*.

[45] Esther Kaufmann and Abraham Bernstein. 2010. Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. *Journal of Web Semantics* 8, 4 (2010), 377–393.

[46] Esther Kaufmann, Abraham Bernstein, and Renato Zumstein. 2006. Querix: A natural language interface to query ontologies based on clarification dialogs. In *5th international semantic web conference (ISWC 2006)*. Citeseer, 980–981.

[47] Anirudh Khatry, Joyce Cahoon, Jordan Henkel, Shaleen Deep, Venkatesh Emani, Avrilia Floratou, Sumit Gulwani, Vu Le, Mohammad Raza, Sherry Shi, et al. 2023. From Words to Code: Harnessing Data for Program Synthesis from Natural Language. *arXiv preprint arXiv:2305.01598* (2023).
[48] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. 2022. Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications* (2022), 1–32.
[49] Dae Hyun Kim, Enamul Hoque, and Maneesh Agrawala. 2020. Answering questions about charts and generating visual explanations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.
[50] Sanghoon Kim and In-Young Ko. 2022. A Conversational Approach for Modifying Service Mashups in IoT Environments. In *CHI Conference on Human Factors in Computing Systems*. 1–16.
[51] Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the Web with Natural Language. In *CHI Conference on Human Factors in Computing Systems*. 1–17.
[52] Yelim Kim, Mohi Reza, Joanna McGrenere, and Dongwook Yoon. 2021. Designers characterize naturalness in voice user interfaces: their goals, practices, and challenges. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–13.
[53] A Baki Kocaballi, Enrico Coiera, and Shlomo Berkovsky. 2020. Revisiting habitability in conversational systems. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–8.
[54] Rafal Kocielnik, Saleema Amershi, and Paul N Bennett. 2019. Will you accept an imperfect ai? exploring designs for adjusting end-user expectations of ai systems. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–14.
[55] Dimosthenis Kontogiorgos and Hannah RM Pelikan. 2020. Towards adaptive and least-collaborative-effort social robots. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. 311–313.
[56] Gierad P Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, Jason Linder, and Eytan Adar. 2013. Pixeltone: A multimodal interface for image editing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2185–2194.
[57] Fei Li and Hosagrahar V Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment* 8, 1 (2014), 73–84.
[58] Toby Jia-Jun Li, Jingya Chen, Haijun Xia, Tom M Mitchell, and Brad A Myers. 2020. Multi-modal repairs of conversational breakdowns in task-oriented dialogs. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 1094–1107.
[59] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M Mitchell, and Brad A Myers. 2019. Pumice: A multi-modal agent that learns concepts and conditionals from natural language and demonstrations. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*. 577–589.
[60] Yulin Li, Yuxi Qian, Yuechen Yu, Xiameng Qin, Chengquan Zhang, Yan Liu, Kun Yao, Junyu Han, Jingtuo Liu, and Errui Ding. 2021. Structext: Structured text understanding with multi-modal transformers. In *Proceedings of the 29th ACM International Conference on Multimedia*. 1912–1920.
[61] Yunyao Li, Huahai Yang, and HV Jagadish. 2005. Nalix: an interactive natural language interface for querying xml. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 900–902.
[62] Yunyao Li, Huahai Yang, and HV Jagadish. 2006. Constructing a generic natural language interface for an XML database. In *International Conference on Extending Database Technology*. Springer, 737–754.
[63] Q Vera Liao, Matthew Davis, Werner Geyer, Michael Muller, and N Sadat Shami. 2016. What can you do? Studying social-agent orientation and agent proactive interactions with an agent for employees. In *Proceedings of the 2016 acm conference on designing interactive systems*. 264–275.
[64] Q. Vera Liao, Matthew Davis, Werner Geyer, Michael Muller, and N. Sadat Shami. 2016. What Can You Do? Studying Social-Agent Orientation and Agent Proactive Interactions with an Agent for Employees. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 264–275. https://doi.org/10.1145/2901790.2901842
[65] Q Vera Liao, Werner Geyer, Michael Muller, and Yasaman Khazaen. 2020. Conversational interfaces for information search. In *Understanding and improving information search*. Springer, 267–287.
[66] Q Vera Liao, Muhammed Mas-ud Hussain, Praveen Chandar, Matthew Davis, Yasaman Khazaeni, Marco Patricio Crasso, Dakuo Wang, Michael Muller, N Sadat Shami, and Werner Geyer. 2018. All work and no play?. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.
[67] James Lin, Jeffrey Wong, Jeffrey Nichols, Allen Cypher, and Tessa A Lau. 2009. End-user programming of mashups with vegemite. In *Proceedings of the 14th international conference on Intelligent user interfaces*. 97–106.
[68] Michael Xieyang Liu, Advait Sarkar, Carina Negreanu, Benjamin Zorn, Jack Williams, Neil Toronto, and Andrew D Gordon. 2023. "What It Wants Me To Say": Bridging the Abstraction Gap Between End-User Programmers and Code-Generating Large Language Models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–31.

[69] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.

[70] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786* (2021).

[71] Ewa Luger and Abigail Sellen. 2016. " Like Having a Really Bad PA" The Gulf between User Expectation and Experience of Conversational Agents. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 5286–5297.

[72] Hao Ma, Michael Lyu, and Irwin King. 2010. Diversifying query suggestion results. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 24. 1399–1404.

[73] Robert C Miller, Victoria H Chou, Michael Bernstein, Greg Little, Max Van Kleek, David Karger, and MC Schraefel. 2008. Inky: a sloppy command line for the web with rich visual feedback. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. 131–140.

[74] Christopher H Morrell. 1998. Likelihood ratio testing of variance components in the linear mixed-effects model using restricted maximum likelihood. *Biometrics* (1998), 1560–1568.

[75] Jesse Mu and Advait Sarkar. 2019. Do we need natural language? Exploring restricted language interfaces for complex domains. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–6.

[76] Omar Mubin, Christoph Bartneck, Loe Feijs, Hanneke Hooft van Huysduynen, Jun Hu, and Jerry Muelver. 2012. Improving speech recognition with the robot interaction language. *Disruptive science and Technology* 1, 2 (2012), 79–88.

[77] Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-Based AMR Parsing with Reinforcement Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 4586–4592. https://doi.org/10.18653/v1/P19-1451

[78] Joe O'Connor and Jacob Andreas. 2021. What Context Features Can Transformer Language Models Use? *arXiv preprint arXiv:2106.08367* (2021).

[79] Fatma Őzcan, Abdul Quamar, Jaydeep Sen, Chuan Lei, and Vasilis Efthymiou. 2020. State of the art and open challenges in natural language interfaces to data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2629–2636.

[80] Florian Pecune, Shruti Murali, Vivian Tsai, Yoichi Matsuyama, and Justine Cassell. 2019. A model of social explanations for a conversational movie recommendation system. In *Proceedings of the 7th International Conference on Human-Agent Interaction*. 135–143.

[81] Kimberly A Pollard, Stephanie M Lukin, Matthew Marge, Ashley Foots, and Susan G Hill. 2018. How We Talk with Robots: Eliciting Minimally-Constrained Speech to Build Natural Language Interfaces and Capabilities. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 62. SAGE Publications Sage CA: Los Angeles, CA, 160–164.

[82] Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. 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)*. 878–888.

[83] Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–7.

[84] Xin Rong, Shiyan Yan, Stephen Oney, Mira Dontcheva, and Eytan Adar. 2016. Codemend: Assisting interactive programming with bimodal embedding. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 247–258.

[85] Diptikalyan Saha, Neelamadhav Gantayat, Senthil Mani, and Barry Mitchell. 2017. Natural language querying in SAP-ERP platform. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. 878–883.

[86] Advait Sarkar, Andrew D Gordon, Carina Negreanu, Christian Poelitz, Sruti Srinivasa Ragavan, and Ben Zorn. 2022. What is it like to program with artificial intelligence? *arXiv preprint arXiv:2208.06213* (2022).

[87] Viktor Schlegel, Benedikt Lang, Siegfried Handschuh, and André Freitas. 2019. Vajra: step-by-step programming with natural language. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 30–39.

[88] Vidya Setlur, Sarah E Battersby, Melanie Tory, Rich Gossweiler, and Angel X Chang. 2016. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th annual symposium on user interface software and technology*. 365–377.

[89] Janet Siegmund, Christian Kästner, Jörg Liebig, Sven Apel, and Stefan Hanenberg. 2014. Measuring and modeling programming experience. *Empirical Software Engineering* 19, 5 (2014), 1299–1334.

[90] Arjun Srinivasan, Mira Dontcheva, Eytan Adar, and Seth Walker. 2019. Discovering natural language commands in multimodal interfaces. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 661–672.

[91] Antonia Tolzin, Anita Körner, Ernestine Dickhaut, Andreas Janson, Ralf Rummer, and Jan Marco Leimeister. 2023. Designing Pedagogical Conversational Agents for Achieving Common Ground. In *International Conference on Design Science Research in Information Systems and Technology*. Springer, 345–359.

[92] Keisuke Tsunoda and Reiko Hishiyama. 2010. Design of multilingual participatory gaming simulations with a communication support agent. In *Proceedings of the 28th ACM International Conference on Design of Communication*. 17–25.

[93] Prasetya Utama, Nathaniel Weir, Fuat Basik, Carsten Binnig, Ugur Cetintemel, Benjamin Hättasch, Amir Ilkhechi, Shekar Ramaswamy, and Arif Usta. 2018. An end-to-end neural natural language interface for databases. *arXiv preprint arXiv:1804.00401* (2018).

[94] Jessica Van Brummelen, Kevin Weng, Phoebe Lin, and Catherine Yeo. 2020. CONVO: What does conversational programming need?. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–5.

[95] Karel van den Bosch, Tjeerd Schoonderwoerd, Romy Blankendaal, and Mark Neerincx. 2019. Six challenges for human-AI Co-learning. In *Adaptive Instructional Systems: First International Conference, AIS 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Orlando, FL, USA, July 26–31, 2019, Proceedings 21*. Springer, 572–589.

[96] Justin D Weisz, Mohit Jain, Narendra Nath Joshi, James Johnson, and Ingrid Lange. 2019. BigBlueBot: teaching strategies for successful human-agent interactions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 448–459.

[97] Terry Winograd. 1971. *Procedures as a representation for data in a computer program for understanding natural language*. Technical Report. MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC.

[98] Christine T Wolf. 2019. Explainability scenarios: towards scenario-based XAI design. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 252–257.

[99] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. 2022. Promptchainer: Chaining large language model prompts through visual programming. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 1–10.

[100] Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In *Proceedings of the 2022 CHI conference on human factors in computing systems*. 1–22.

[101] Frank F Xu, Bogdan Vasilescu, and Graham Neubig. 2022. In-ide code generation from natural language: Promise and challenges. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 2 (2022), 1–47.

[102] Bowen Yu and Cláudio T Silva. 2019. FlowSense: A natural language interface for visual data exploration within a dataflow system. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 1–11.

[103] JD Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–21.

[104] Jennifer Zamora. 2017. I'm sorry, dave, i'm afraid i can't do that: Chatbot perception and expectations. In *Proceedings of the 5th international conference on human agent interaction*. 253–260.

[105] Yiqian Zou. 2020. An experimental evaluation of grounding strategies for conversational agents. (2020).

A Statistical Model Summary

We summarize full model outputs of linear mixed-effects regression models that showed significance.

Table 3. Speaker change costs

Likelihood ratio test					
Effect		logLik		χ^2	p
Condition		-374.1		13.47	**0.004*****
Fixed Effects					
	Estimate (β)	SE	95% CI		t
Intercept	6.50	0.65	5.18	7.82	9.98
Grounding	1.10	0.86	-0.62	2.82	1.27
Multiple grounding	0.48	0.86	-1.24	2.19	0.55
Structured grounding	-2.03	0.86	-3.74	-0.31	-2.34
Random Effects					
				Variance	S.D.
Task ID				0.10	0.32
Participant ID				6.19	2.49
Residual				2.58	1.61

Table 4. Task success rate

Likelihood ratio test					
Effect		logLik		χ^2	p
Condition		-90.2		9.668	**0.022****
Fixed Effects					
	Estimate (β)	SE	95% CI		t
Intercept	0.23	0.10	0.03	0.42	2.33
Grounding	0.13	0.13	-0.13	0.38	0.97
Multiple grounding	0.13	0.13	-0.13	0.38	0.97
Structured grounding	0.40	0.13	0.14	0.66	3.10
Random Effects					
				Variance	S.D.
Task ID				0.00	0.05
Participant ID				0.12	0.34
Residual				0.10	0.31

Received July 2023; revised April 2024; accepted July 2024