

ONYX: Assisting Users in Teaching Natural Language Interfaces Through Multi-Modal Interactive Task Learning

Marcel Ruoff

marcel.ruoff@kit.edu

Karlsruhe Institute of Technology
Karlsruhe, Germany

Brad A. Myers

bam@cs.cmu.edu

Carnegie Mellon University
Pittsburgh, USA

Alexander Maedche

alexander.maedche@kit.edu

Karlsruhe Institute of Technology
Karlsruhe, Germany

ABSTRACT

Users are increasingly empowered to personalize natural language interfaces (NLIs) by teaching how to handle new natural language (NL) inputs. However, our formative study found that when teaching new NL inputs, users require assistance in clarifying ambiguities that arise and want insight into which parts of the input the NLI understands. In this paper we introduce ONYX, an intelligent agent that interactively learns new NL inputs by combining NL programming and programming-by-demonstration, also known as multi-modal interactive task learning. To address the aforementioned challenges, ONYX provides *suggestions* on how ONYX could handle new NL inputs based on previously learned concepts or user-defined procedures, and poses *follow-up questions* to clarify ambiguities in user demonstrations, using *visual and textual aids* to clarify the connections. Our evaluation shows that users provided with ONYX's new features achieved significantly higher accuracy in teaching new NL inputs (median: 93.3%) in contrast to those without (median: 73.3%).

CCS CONCEPTS

- Human-centered computing → User interface programming; Natural language interfaces.

KEYWORDS

Interactive Task Learning, End User Development, Natural Language Interfaces, Data Visualization Tools

ACM Reference Format:

Marcel Ruoff, Brad A. Myers, and Alexander Maedche. 2023. ONYX: Assisting Users in Teaching Natural Language Interfaces Through Multi-Modal Interactive Task Learning. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/3544548.3580964>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI '23, April 23–28, 2023, Hamburg, Germany

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-9421-5/23/04...\$15.00

<https://doi.org/10.1145/3544548.3580964>

1 INTRODUCTION

With the recent advances in natural language processing, users are increasingly provided with the ability to interact through natural language (NL) with their smartphones (e.g., Siri; Google Assistant), the web (e.g., FireFox Voice [8]) or specific applications like data visualization tools (e.g., Power BI Q&A; Tableau's Ask Data). While these NL interfaces (NLIs) initially pursued a one-size-fits-all design, developers soon realized that different users or contexts require supporting more personalized NL inputs [14] since users are otherwise often quitting those NLIs instead of retrying after they face breakdowns [22]. Therefore, users are able to teach some existing NLIs how to handle new NL inputs in limited ways. For example, NLIs can be taught by end users to perform custom procedures combining multiple functionalities of the underlying system (*procedural knowledge*), such as the custom commands provided by Apple's Siri [2]. Another example is how end users can provide synonyms to existing concepts or define new concepts (*declarative knowledge*) to improve the NLI's understanding (e.g., what constitutes a "crucial" customer in Power BI Q&A [28]).

To empower end users in extending the previously mentioned NLIs, end user development techniques, such as visual programming (e.g., Siri [2]; Google Assistant [1]) or simpler form-filling techniques (e.g., Power BI Q&A [28]), are utilized to lower the barriers. While visual programming is currently the key approach, it has been shown over decades that creating programs with visual programming languages is still relatively complex for most users [6, 11, 15, 30]. First, end users struggle to select the correct visual programming blocks from the extensive options to create their intended program [6]. Second, detecting and fixing errors in their visual programs is still a stumbling block [18, 30]. A promising approach to address these challenges is interactive task learning (ITL). ITL-based systems do not require end users to search for correct visual programming blocks or familiarize themselves with a programming language. Instead, ITL-based systems combine NL programming and programming-by-demonstration to learn from multi-modal user demonstrations of the task in the actual system [20].

A fundamental challenge in ITL-based systems is their ability to not only learn macro recordings of specific user demonstrations but to be able to *generalize* the derived knowledge to support users in performing *similar* tasks. While previous research on ITL has improved the NLIs' ability to generalize *declarative* knowledge (i.e., concepts such as *hot* and *cold* [25] and values such as customer names [19, 21]) and utilize previously defined declarative knowledge during future demonstrations [25], research opportunities still

remain in generalizing and reusing *procedural* knowledge. Improving the generalization of procedural knowledge is crucial since ambiguities contained in demonstrations through direct manipulation either lead to a narrow understanding of the demonstrated task or require many demonstrations of the same task [46]. Additionally, end users want to build on existing user-defined procedures and therefore need insight into which parts of a new NL input the NLI understands and can already handle.

Therefore, we introduce *ONYX*, an intelligent agent that is able to learn both procedural and declarative knowledge through ITL. Essential capabilities of *ONYX* are its ability to generalize procedural knowledge, and its ability to provide insight into existing declarative and procedural knowledge during the teaching of new NL inputs. *ONYX* learns both from users' direct manipulations (programming-by-demonstration) and NL inputs (NL programming) after encountering a new NL input. Three key novel aspects of *ONYX*'s design are the (i) *suggestions*, (ii) *follow-up questions*, and (iii) *guidance through visual and textual aids* provided by *ONYX*. First, through *suggestions* *ONYX* describes how it can handle new NL inputs based on previously learned concepts and user-defined procedures. Second, *follow-up questions* are utilized to accurately abstract and generalize procedural knowledge by clarifying possible ambiguities in direct manipulation demonstrations by end users. Third, to provide users with *guidance* at crucial stages of the demonstration process, *ONYX* provides visual and textual aids, such as connecting the concepts *ONYX* understood in the new NL input to their associated visual elements in the GUI (i.e., buttons and data fields).

We demonstrate *ONYX*'s capabilities in a custom-built data visualization tool since data visualization tools (1) facilitate complex actions which possibly exhibit ambiguities [46], (2) have a wide range of NL inputs users want to utilize while little labeled NL input exists to train these NLIs to understand this variety [39], and (3) the expressiveness of current learning approaches of new NL inputs is limited (e.g., [28, 41]). We integrated a dataset about the COVID-19 pandemic [10] since end users are familiar with this data and the possible insights they might want to derive.

In developing *ONYX*, we took a user-centered design approach using iterative participatory design with 10 participants to explore what issues end users face in NLIs with ITL capabilities and to derive new designs for *ONYX* to address these challenges. Over the course of four months, we performed six iterations of *ONYX* with 2 - 4 participants per iteration. Each participant took part in two consecutive iterations so we could get feedback from them both when they have minimal knowledge of *ONYX* in their initial iteration, and in the subsequent iteration where they have a deeper understanding.

After building *ONYX* based on that feedback, we performed an online summative evaluation with 42 participants. Further, a think-aloud study with 5 participants provided evidence for the effectiveness and usability of our final design and offered additional details on how users utilize our novel features.

To summarize, the contributions of this paper are:

- (1) A multi-modal ITL approach to enhance existing NLIs iteratively through programming-by-demonstration and NL programming, with the following major advantages:

- (a) **Suggestions provided during the demonstration process** give users insight into how the ITL agent could handle new NL inputs based on previously learned concepts or user-defined procedures and enable users to focus on concepts and procedures that are currently unknown to the ITL agent.
- (b) The ITL agent uses **follow-up questions** to clarify ambiguities in the user's direct manipulation demonstrations to facilitate the abstraction and generalizability of the derived procedural knowledge.
- (c) Users receive a **display of the ITL agent's understanding** of the new NL input grounded in known concepts and visually tied to GUI elements to provide users a deeper understanding of the ITL agent's declarative knowledge.
- (2) The *ONYX* system: an implementation of the aforementioned approach, along with a formative study ($n = 10$) highlighting issues end users face in existing NLIs with ITL capabilities, along with an online experiment ($n = 42$) and think-aloud study ($n = 5$) summatively evaluating its effectiveness and usability. The final evaluation shows that users provided with *ONYX*'s suggestions and follow-up questions achieved significantly ($p < 0.001$) higher accuracy in teaching new NL inputs (median: 93.3%) in contrast to those without (median: 73.3%).

2 RELATED WORK

In this section, we discuss prior work in three areas related to our *ONYX* system: ITL in general, NLIs with learning capabilities, and NLIs for data visualization tools.

2.1 Learning Tasks through Demonstrations

Interactive Task Learning enables end users to automate tasks without requiring them to write code. Instead, users demonstrate the actions required to complete the task, similar to how they would perform the task without ITL. These demonstrations are then utilized to extract the procedural (i.e., relevant actions) and declarative knowledge (i.e., concepts and values) to create a script. Extensive possibilities of tracking and learning from demonstrations (e.g., through APIs [23, 27] or through internal data of the system [3, 19, 44]) have been used in diverse application areas including the creation of GUIs [31, 44] or information visualizations [46], and the automatization of tasks on mobile phones [23, 36], on the web [12, 21], or with robots [40, 42].

A major challenge in all application areas is the generalization of the underlying procedural and declarative knowledge to support users in performing *similar* tasks to those demonstrated. Existing systems focus on generalizing the declarative knowledge by parameterizing the utilized values and concepts either by involving users during the demonstration process (e.g., [12, 19]) or automatically by the system based on internal knowledge (e.g., underlying datasets [21]). However, besides generalizing declarative knowledge, it is also crucial to generalize procedural knowledge. Most importantly, direct manipulations, the main source currently for demonstrations in ITL-based systems, only communicate *what* a user does and not *why* or *how* to perform these actions in varying contexts. Therefore, possible ambiguities in the direct manipulation

need to be clarified to accurately generalize the scripts by deriving the underlying procedural knowledge [25, 46]. APPINITE [24] aims to address this challenge by involving users to clarify ambiguities. Specifically, after APPINITE detects an ambiguity it allows users to describe their intention through NL. However, solely relying on end users to clarify the ambiguities is risky as end users have problems expressing in NL the correct conditions without assistance from the system [9]. Hence, we investigate how end users can be assisted by ONYX in clarifying ambiguities to abstract direct manipulation demonstrations by answering follow-up questions asked by ONYX that only require users to choose among possible interpretations of that ambiguity.

2.2 Natural Language Interfaces with Learning Capabilities

Learning new NL inputs is an important application for ITL since NL allows end users to easily perform complex tasks (i.e., conditional tasks [25]), automate frequently performed tasks [3, 23], access infrequently used actions [17], or provide more natural phrasings for existing NL inputs [45].

To assist end users in teaching new NL inputs, end users require insight into which parts (e.g., words, phrases, and concepts) of a new NL input the NLI understands and can already handle. Most NLIs with learning capabilities, however, do not provide insights into existing declarative (i.e., concepts) and procedural (i.e., actions) knowledge during the demonstration process (e.g., [5, 17, 23, 45]). In contrast, PUMICE [25] provides insight into previously learned declarative knowledge like the concepts *hot* or *cold* during the demonstration process through a multi-turn conversation. However, PUMICE only utilized existing procedural knowledge (i.e., user-defined procedures) if PUMICE has an understanding of how to perform the complete procedure but does not provide insights if it has a partial understanding of what actions it should perform. Providing insight into procedural knowledge differs from insights into declarative knowledge. While PUMICE has to clarify declarative knowledge which can be categorized into boolean and value conditions with their corresponding concepts and values and hence can be easily parsed into a depth-first fashion [25], ONYX has to provide insights into procedural knowledge that can consist of a previously unknown number of actions which can build on top of each other or are completely independent and hence it is difficult for the system to define the relation beforehand. Similarly to PUMICE, AutoVCI [35] enables users to teach synonyms for the NL inputs utilizing existing procedural knowledge for complete NL inputs. In addition, ONYX allows the combination of previous procedural knowledge of multiple existing NL inputs to inform the teaching of new NL inputs through suggestions.

2.3 Natural Language Interfaces for Data Visualization Tools

NLIs have been increasingly utilized to assist users in analyzing and exploring data in data visualization tools (e.g., [13, 16, 26, 37, 38]). Previous studies showed that extending data visualization tools through NLIs particularly helps users perform tasks that would otherwise require multiple adjustments in the GUI [37] or complex filter settings [16]. However, while the variety of use cases has

grown over the last decades, a large gulf between user expectations and the capabilities of NLIs still exists [43].

The major challenge of NLIs in data visualization tools is that users expect the NLI to understand a wide range of NL inputs to create and adapt the data visualizations [39], and expect the NLIs to have a deep understanding of the context and dataset the users are currently working with [43]. When current NLIs for data visualization tools fail to understand the users' goal of an NL input they either prompt users to retry their action differently [16, 38] or involve the user to clarify possible misunderstandings in the NL input [13, 32, 37]. While the latter already improves the user experience of NLIs for data visualization tools [13], existing systems do not learn from these demonstrations for future interactions except for simple form-filling techniques (e.g., [28]) and therefore continuously require end users to clarify the same misunderstandings.

Hence, we investigate ONYX in the context of data visualization tools to showcase how an NLI can learn from demonstrations to increasingly improve its coverage of the NL inputs users want to utilize.

3 FORMATIVE STUDY & DESIGN GOALS

We took a user-centered approach [29] to address issues end users face in existing NLIs with ITL capabilities by using participatory design. We recruited 10 participants (8 males; 2 females; M = 27.5 years SD = 11.4) over the course of 4 months. Each participant took part in two sessions one week apart (about 1 hour per session). In each session, participants first completed target replication tasks and afterward an open-ended data exploration to additionally understand how ONYX impacts users' analytic flow. We recorded their think-aloud statements, coded them, and derived (among others) the following insights.

The full findings of this study have been previously published as a poster¹, but we summarize the most relevant insights from the formative study for the design of ONYX next. From these insights, we distilled six design goals (DGs) for enabling effective ITL for NLIs.

3.1 Understanding of the NLI's Existing Knowledge

In the earlier stage of the participatory design process, while users generally understood *how* they could demonstrate the meaning of an NL input to the system due to an initial introduction by the supervising researcher and a textual introduction by the ITL agent after the breakdown, they did not know *what* they need to teach ONYX. Especially because ONYX's error messages were generic, like in many other NLIs [4]. Participants, therefore, tried to derive the NLI's existing understanding of their NL input during the demonstration process in a trial-and-error approach by changing the NL input incrementally and paying attention to whether ONYX understood this adapted NL input as part of its NL programming capabilities. However, this caused significant disruptions to their analytic flow.

¹To preserve the anonymity of this submission, in the supplementary material we provide an anonymized version of that paper that describes the formative study in more detail.

DG1. Be specific about what parts of the NL input the NLI understood and did not understand.

Participants utilized this trial-and-error approach to get insight into both the (i) procedural and (ii) declarative knowledge the NLI possesses. First, when *ONYX* failed to understand a joined NL input (e.g., Remove Deaths and focus on Cases or Show states **with mhighlighted mhighlights**) some participants would enter the parts of the NL input during the demonstration separately to check whether *ONYX* possesses procedural knowledge for parts of the NL input.

DG2. Provide suggestions based on the parts of the NL input the NLI understood.

Further, participants were often unsure whether *ONYX* failed due to missing declarative or procedural knowledge. For example, in the NL input Give me TX *ONYX* could either lack procedural knowledge for how to handle **Give me [something]**, lack declarative knowledge that TX is an abbreviation for Texas or both. Hence, participants were comparing parts of the NL inputs with labels in the GUI elements to reassure themselves which concepts they thought *ONYX* understood, and which it did not.

DG3. Ground the NLI's declarative knowledge for parts of the NL input through visual and textual aids utilizing the GUI elements.

3.2 Ambiguous Direct Manipulation Demonstrations

Participants mostly utilized direct manipulations for their demonstrations, similar to users in other ITL-based systems [23]. Hence, abstracting and generalizing the direct manipulation demonstrations is especially important to derive an accurate script.

However, participants realized that when ambiguities occur in direct manipulation interactions, *ONYX* occasionally chose the wrong interpretation of their demonstration. In our initial version, participants appreciated the ability to directly edit a visual representation of the script during the demonstration as this simplified changes and deletions in the script if they noticed misunderstandings. Participants additionally acquired an understanding of the current interpretation of their actions by *ONYX* through this visual representation.

DG4. Enable users to edit the ITL agent's understanding of the demonstrations performed.

However, sessions using early versions highlighted two shortcomings of solely relying on this visual representation of the script for addressing misunderstandings. First, if users did not notice the incorrect interpretation of the ambiguous direct manipulation they would not try to clarify. Second, if users did notice the incorrect interpretation, they would only realize that the interpretation is incorrect, but not what caused this unwanted behavior. Hence, earlier sessions reveal the need for *ONYX* to actively notify users of ambiguities and subsequently describe the possible interpretations for users to choose from.

DG5. Address ambiguities in direct manipulation demonstrations through follow-up questions.

3.3 Design of Assistance

After addressing the challenges mentioned above by assisting users through *suggestions*, *follow-up questions*, and additional *visual and textual aids*, the subsequent iterations of user studies highlighted the positive effect of these features on lowering the disruption of participants' analytic flow. However, participants additionally **with mhighlighted mhighlights** trade-offs with the design of this assistance:

Timing of Assistance: While asynchronously providing assistance would minimize disruptions (e.g., at the end of the demonstration process), synchronous assistance helps users better understand the information *ONYX* requires to learn how to accurately handle the new NL input.

Modality of Assistance: While visually presenting follow-up questions as GUI elements helps users make fast decisions, users can better understand the ambiguity *ONYX* is trying to address when *ONYX* provides textual follow-up questions.

In our context, we opted to have *ONYX* use synchronous assistance and textual follow-up questions. Additional iterations of participatory design highlighted that a deep understanding of the information *ONYX* requires is more urgent than being incrementally faster. Furthermore, the disruptions of synchronous assistance were minor due to additional visual and textual aids that guide end users in the utilization of the assistance.

DG6. Guide users' attention during assistance through visual and textual aids to minimize the disruption caused by the interruption.

4 ONYX

In this section, we describe how *ONYX* incorporates the previously articulated design goals. Specifically, we describe an example scenario that illustrates how users can personalize the NLI. Subsequently, we detail how *ONYX*: (i) learns from multi-modal demonstrations, (ii) derives suggestions, (iii) identifies ambiguities in the demonstrated actions, (iv) provides visual and textual aids, and (v) generalizes the derived scripts.

4.1 Example Scenario

Mikki, a student from North Carolina recently moved to Pennsylvania for her studies. She aims to utilize *ONYX*, which integrates a dataset [10] about the COVID-19 pandemic in the United States of America. We will utilize this dataset throughout this paper to provide consistency.

To get an understanding of the course of the pandemic in her home state, Mikki starts with a scatterplot visualization that depicts deaths on the y-axis, fully vaccinated on the x-axis, and a color encoding based on the dates since the beginning of the pandemic filtered for North Carolina (see Figure 1 A). To focus on some points of interest she enters the NL input Focus on those with less **than 1 million fully vaccinated** into the NLI. While *ONYX* would understand other NL inputs for this goal, it does not yet understand how to handle the NL input that Mikki used. However, Mikki would prefer to use her own NL input as it feels more natural to her. After deciding to teach *ONYX*, *ONYX* provides feedback in the NLI and provides in its training interface an explanation of *ONYX*'s understanding of the NL input (DG1 & DG3). Additionally, *ONYX*

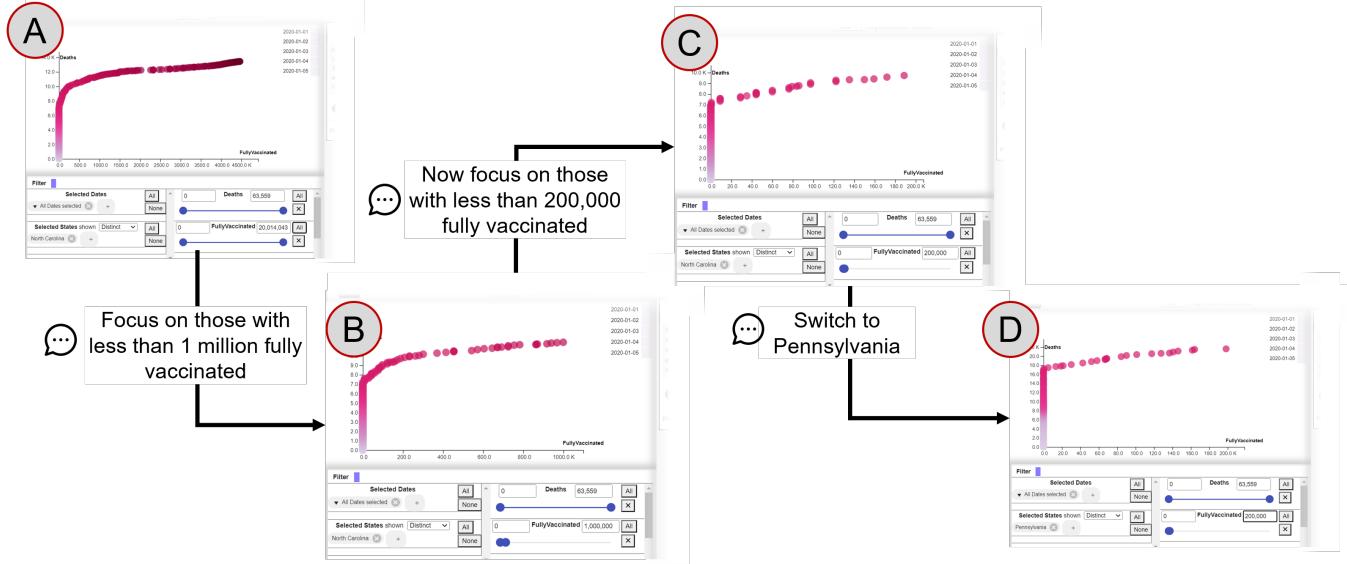


Figure 1: Data Visualizations and NL Inputs utilized in the Scenario.

provides suggestions on how to handle parts of the NL input for Mikki to reuse and build on (DG2). In this case, ONYX reuses procedural knowledge for **less than 1 million fully vaccinated**. The changes are directly carried out in the data visualization tool (see Figure 2 (C)) and added to the *current understanding* section of the UI (see Figure 2 (E3)). ONYX further gives a brief explanation in the NLI on which part of the NL input the suggestion is based on (see Figure 2 (D)). If the suggestion did not fit Mikki's understanding, she could refine or delete the suggestion (see Figure 2 (E3)). However, since the suggestion fits her understanding, she checks the visual representation of the script and finishes the demonstration as the visual representation of the script already fits her understanding of the NL input. ONYX immediately responds that this NL input is now available to her.

She wants to focus on the vaccination roll-out and therefore enters (NL) Now focus on those **with less than 200,000 fully vaccinated**. Due to her previous demonstration, ONYX knows how to handle this NL input and adapts the data visualization accordingly (see Figure 1 (C)).

Now Mikki wants to know if Pennsylvania has a similar trend. Hence, Mikki enters (NL) Switch to **Pennsylvania**. Again, ONYX does not know how to handle the NL input and therefore initiates the demonstration mode (see Figure 3 (A)). After deciding to demonstrate the NL input, Mikki directly removes **North Carolina** from the current selection of states displayed in the visualization. This triggers an ambiguity in the back-end of ONYX since ONYX is unsure if Mikki wanted to (i) specifically delete **North Carolina** or (ii) if *all* states should be removed from the selected states. This is crucial since Mikki might later use this NL input again with different or even multiple states selected. Hence, ONYX prompts a follow-up question in the NLI which asks whether Mikki wanted to remove all states from selected states as a yes/no question (DG5

& DG6) (see Figure 3 (B)). After reading the question, she quickly selects Yes as the appropriate answer and ONYX adapts its understanding in response (see Figure 3 (C)). Mikki performs the rest of the actions and finishes the demonstration mode to which ONYX responds that the action is now available to her (see Figure 3 (E)).

Mikki might now use similar NL inputs during further exploration with different parameters in the NL input such as different values in the numeric relation of the first NL input or different states instead of Pennsylvania in the second NL input.

4.2 Key Design Features

4.2.1 Learning from Multi-Modal User Demonstrations.

To allow users to demonstrate actions using both direct manipulation and NL input, ONYX utilizes programming-by-demonstration and NL programming to translate both types of interactions into a common script. One way that users are able to adapt the script is by adapting its visual representation (see Figure 2 (E3)) by selecting a different keyword in the display of an action to correct minor misunderstandings (DG4) (see Figure 4 (B)). Major misunderstandings can be addressed by deleting the incorrect parts of the script and redoing them. To continuously check whether the goal of the NL input has been attained, ONYX compares the concepts the user utilized in the script with the ones extracted as parameters from the initial NL input. If all parameters have been utilized, then ONYX inquires whether the user has completed the demonstration (see Figure 4 (A)).

4.2.2 Suggestions.

ONYX provides suggestions based on procedural knowledge associated with (i) *existing user-defined procedures* and (ii) *known concepts utilized in the NL input*. First, suggestions can be triggered if parts of the NL input are similar to known NL inputs and fulfill the requirements of its associated user-defined procedures (see Figure 5

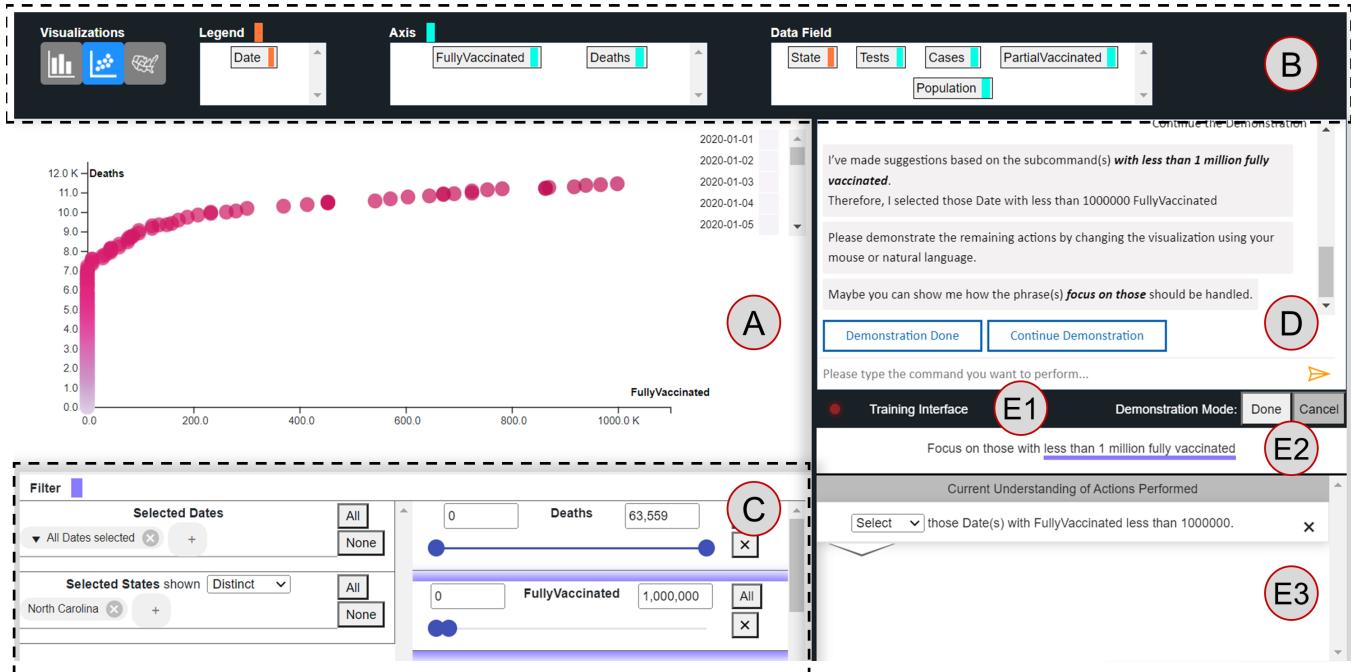


Figure 2: User Interface of the Data Visualization Tool with integrated ITL-based NLI during the Demonstration Process. (A) Visualization canvas, (B) Buttons to adapt chart type and encodings, (C) Filter pane to provide constraints, (D) NLI providing text input and feedback, (E) Signalling indicator and buttons to finish the demonstration mode, (E2) NL input to be demonstrated, (E3) Visual Representation of the Script.

(B). Second, to provide insightful suggestions when only a few user-defined procedures exist, ONYX additionally provides suggestions based on known concepts. Only numeric filters and chart types can trigger suggestions (see Figure 5 (A)) as they are associated with one specific type of procedural knowledge. Concepts such as dimensions, on the other hand, can be associated with multiple (e.g., filtering, selecting/removing as axis, etc.) and hence could lead to incorrect suggestions.

After receiving the NL input from the NL Parser and its associated information (i.e., dependency tree, numeric relations, and named entities), ONYX searches in a depth-first manner for parts of the NL input ONYX knows how to handle (DG2) (see Figure 5). It saves the suggestions together with neighboring parts of the NL input that need to be demonstrated. ONYX provides these suggestions in the order they occur in the NL input. If not actively requested by users, ONYX only provides the next suggestion after the parameters contained in the neighboring parts of the NL input are utilized in the users' demonstrations.

ONYX makes a suggestion by performing the associated actions directly in the data visualization tool and updates the script and the associated visual representation (see Figure 4 (B)). Furthermore, in the NLI, ONYX provides a short explanation of which part of the NL input its suggestion is based on, what actions it performed, and which parts of the NL input associated with the suggestion it did not understand (see Figure 4 (A)).

In the example depicted in Figure 5, ONYX provides the first suggestion regarding the numeric relation (see Figure 5 (A)). It requests the user to demonstrate what the component **Show me all states** means because it recognized that these components are connected. The suggestion in Fig. 5 (B) is only provided after ONYX recognizes that the user demonstrated something connected with the concept "states" (i.e., selecting a state filter). Users can also actively request additional suggestions by utilizing the NLI (e.g., by entering **Next suggestion** or **Help me**).

4.2.3 Follow-Up Questions.

Follow-up questions are used in ONYX to abstract the meaning of low-level direct manipulations to a higher-level understanding by addressing possible ambiguities. To detect these ambiguities in the users' demonstrations, ONYX utilizes conditions based on the parameters extracted from the NL input, the current script, and the demonstration performed (DG5). The conditions integrated into the current instantiation of ONYX are able to detect (i) *direct manipulation demonstration ambiguities*, such as whether users wanted to remove the specific state manipulated or all states if the selected states are empty afterward and (ii) *language ambiguities*, such as if states in **Show me all states** refers to the filter or also to the legend of the data visualization.

Direct manipulation demonstration ambiguities are only triggered if ONYX decides that it can not clarify the ambiguity on its own by utilizing information from the articulated NL input to preserve

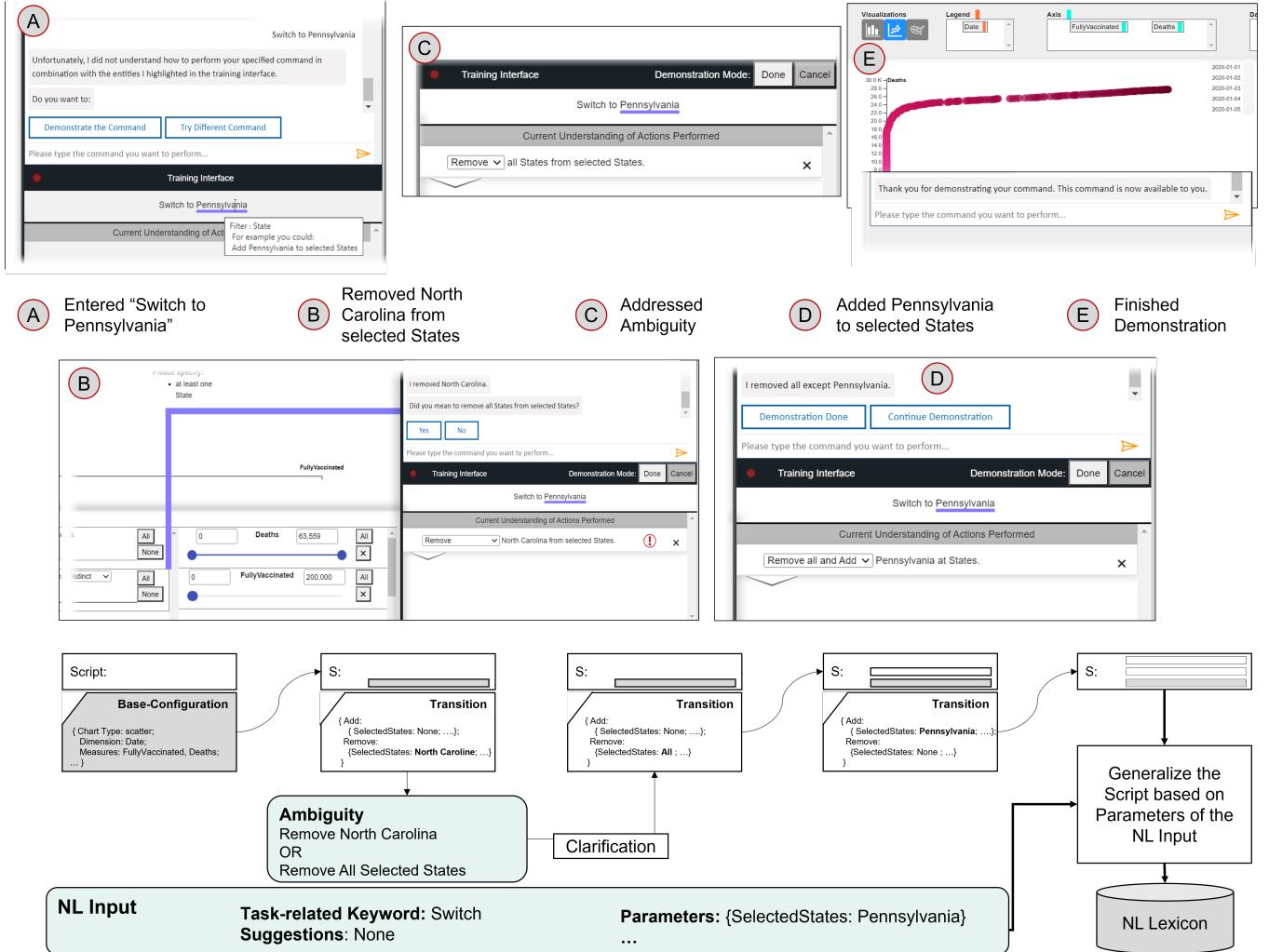


Figure 3: Progression of the User Interface, the Script, and the ITL Agent during the Training for the NL Input `Switch to Pennsylvania`.

users from unnecessary interruptions. For example, when ONYX is unsure whether users only want to remove a specific state or all states, ONYX assumes the former is the correct interpretation without involving users if the removed state was utilized in the NL input.

If ONYX decides that the user is required to abstract the meaning of the direct manipulation demonstration, then ONYX directly asks the follow-up question after one of the conditions is triggered (synchronous assistance) (see Figure 6 (C)). Users can directly address the ambiguity by selecting Yes or No in the NLI or they can continue the demonstration process and address the ambiguity in the visual representation of the script. If multiple conditions for ambiguities are triggered, then ONYX prompts the next follow-up question only after the previous one is clarified to avoid information overload.

4.2.4 Visual and Textual Aids.

Aids for the NL Input. To provide users at the beginning of the

demonstration process an understanding of why the NLI failed, ONYX utilizes the extracted procedural and declarative knowledge to highlight the parts it did understand (DG1). Users can request a visual and textual aid by hovering over the underlined named entities (i.e., measures, dimensions, and categorical filters) and numeric relations of the NL input (see Figure 7 (A)). Upon hovering, ONYX provides a short explanation of its declarative knowledge (e.g., that it recognizes it as a filter) and exemplary actions based on the current context of the data visualization tool (procedural knowledge) as a textual aid. Furthermore, ONYX provides visual aids to ground these concepts in the GUI by highlighting the corresponding GUI elements, such as filter panes or labels (see Figure 7 (A)) (DG3).

Aids for Follow-up Questions. Visual aids for follow-up questions are requested by users either by hovering over the follow-up question in the NLI (see Figure 6 (C)) or by hovering over the GUI element that triggered the ambiguity (see Figure 6 (A)) (DG1). ONYX guides

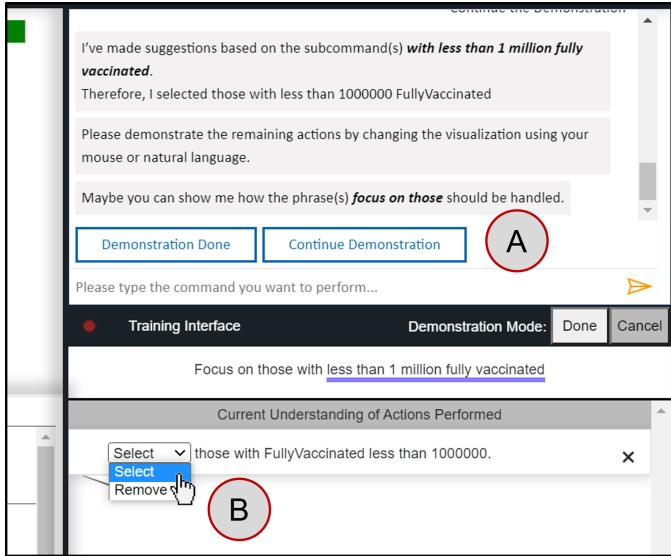


Figure 4: Screenshot of the UI when ONYX detects a possible Goal Attainment during the Demonstration Process.

the attention of users by connecting the GUI element associated with the ambiguity to the follow-up question with a color-coded line (see Figure 6 (B) (DG6)).

4.2.5 Generalization.

ONYX is able to learn knowledge on three levels of generalizability by utilizing the named entities and numeric relations to parameterize both the derived script and the associated NL input. First, verbs (e.g., **Giving [something]**/ **Focusing on [something]**) and their connected procedural knowledge are generalized across various datasets and are only dependent on the functionality provided by the GUI. Second, ONYX links named entities (e.g., States) and relations to abstract concepts specific to data visualizations (e.g., Dimensions, Measures) (see Figure 5). The ITL agent only utilizes the abstract concept for deriving the possible values for parameters in the NL input and is, hence, able to generalize NL inputs across different datasets. However, currently, a JSON-File must be changed manually that maps data fields in the data set to their abstract concepts (e.g., `{'Measures': ['Deaths', ...], 'Dimensions': ['States', ...]}`). Third, ONYX is able to learn a narrow understanding of concepts that are connected to a specific data field (e.g., **southern** states, **soaring** number of deaths) and therefore would, e.g., support asking about southern states in other datasets, but not southern countries. This is due to the fact that ONYX connects this to a well-defined condition (e.g., southern states are Texas, Mississippi, ... / soaring refers to values higher than ...).

4.3 System Architecture

The ONYX system employs a web-based, client-server model. It utilizes HTML5, CSS3, and JavaScript for its web-client and Python for the access and processing of the dataset. The interface manager coordinates the communication between the user interface of ONYX and its NL Parser and ITL agent. The data visualization tool of ONYX

is built on the D3.js library [7] and supports bar charts, scatterplots, and map charts as visualizations as well as categorical and numeric filters.

Figure 8 depicts ONYX's architecture with its three main components: (i) Interface Manager, (ii) NL Parser, and (iii) ITL agent.

4.3.1 Natural Language Parser.

The NL input entered in the NLI is forwarded to an NL Parser. First, the NL Parser utilizes the results from Microsoft's Language Understanding and Intelligent Service (LUIS) as well as the parts of speech tags and the dependency tree from Google Cloud Natural Language API to extract named entities and relations. Subsequently, the NL Parser identifies whether the users' NL input is targeted at (i) *adapting the visualization* (e.g., `(Switch to Pennsylvania)`) or at (ii) *interacting with the ITL agent* (e.g., `(Finish the demonstration)`) by utilizing LUIS intent classification that is fine-tuned on examples extracted from the formative study and additional examples generated by the researchers. If the NL input is targeted at adapting the data visualization, the NL Parser applies a lexicon-based approach utilizing a Bigram Dependency Kernel [34] to derive the associated script from the NL Lexicon. The NL Lexicon is instantiated as a table consisting of entries for all previously trained NL inputs specifying (i) the parameterized trained NL input as an index, (ii) a parameterized bigram-representation of the NL input, (iii) the associated parameterized procedural knowledge, (iv) the required concepts that need to be included in the NL input, and (v) the ID of the user who taught the NL input or whether it is a foundational NL input provided by developers. Subsequently, the NL Parser augments the generalized script with the previously extracted named entities and relations and forwards it to the Interface Manager to execute the augmented script. If the NL Parser does not find an associated script in the NL Lexicon for the articulated NL input, the ITL agent gets triggered.

4.3.2 Interactive Task Learning Agent.

The ITL agent of ONYX utilizes the internal state of the data visualization tool to access user demonstrations. If the demonstration mode is active, the ITL agent derives a script from the continuous user demonstrations. Through this unified script, ONYX is capable of learning how to handle NL inputs that can also be achieved through direct manipulation. After completing the demonstration mode, the ITL agent checks whether named entities or numeric relations in the NL input also have been utilized in the derived script. The detected instances are then parameterized in the derived script and connected to the named entity or numeric relation in the NL input that has been used for parameterization to enable ONYX to select the right parameter during future usage. The NL input is then stored as a new entry in the NL Lexicon.

5 EVALUATIVE USER STUDIES

We conducted an online experiment and a think-aloud study to evaluate the effectiveness of ONYX. We deliberately chose to initially provide ONYX with only a small set of existing procedural knowledge that covered manipulating all GUI elements. In this, our goal is to show that even with this limited set of known concepts and user-defined procedures, users are still effectively assisted by ONYX in accurately demonstrating new NL inputs.

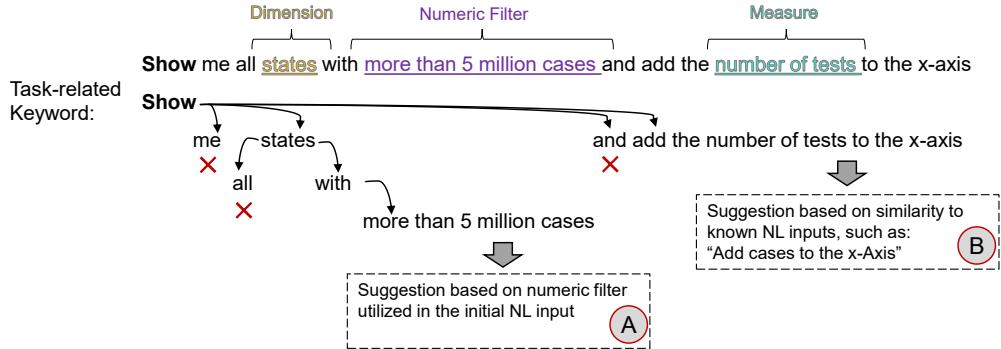


Figure 5: A Sample Sentence tagged and processed by ONYX to derive Suggestions. The Words of the Sentence are connected with directed Arrows based on their Dependency Structure.

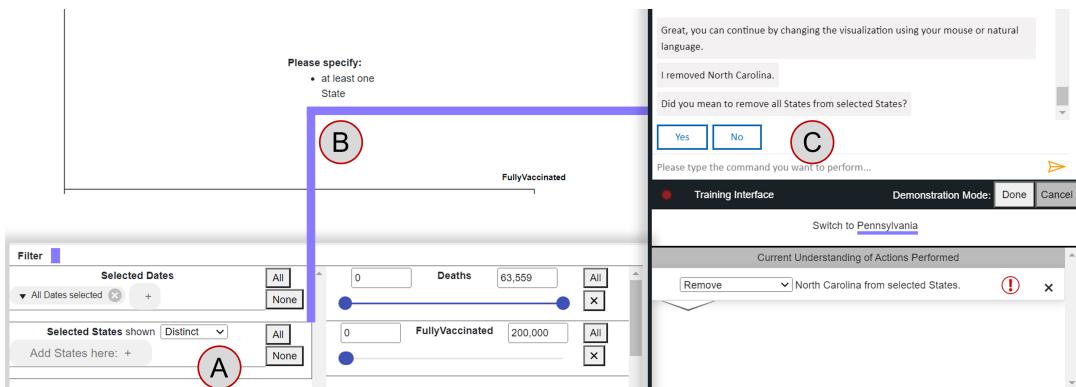


Figure 6: Follow-Up Question, after ONYX detects a Direct Manipulation Ambiguity in the last User Action.

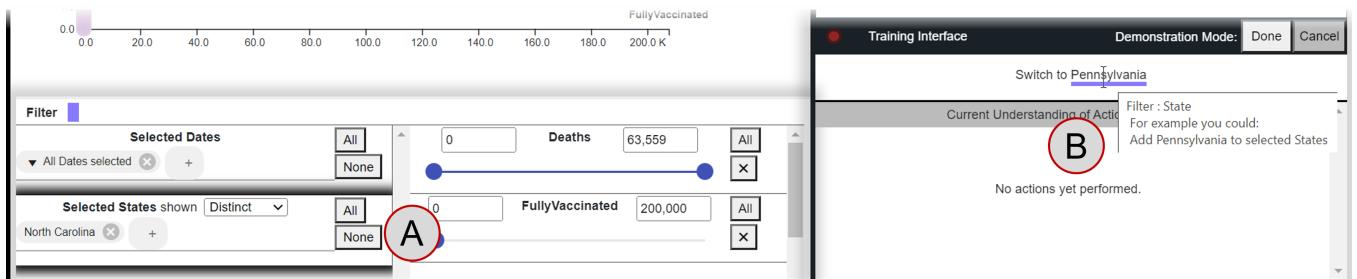


Figure 7: UI when Users hover over underlined Parameters during the Demonstration Process, with highlighted States Filter at A.

In both studies, participants were asked to address three breakdowns of ONYX by demonstrating the meaning of the NL input through direct manipulation and/or NL inputs. With the online experiment, we aimed to provide evidence for the effectiveness of the design goals instantiated in ONYX by utilizing a version of ONYX *without* the suggestions (**DG2**) and without the follow-up questions (**DG5 & DG6**) as a baseline condition as we could not

find similar tools that support users with little programming expertise to complete the target tasks. The baseline version of ONYX still includes the ability to learn from NL inputs (**DG1**: see Section 4.2.1) and basic visual and textual aids (**DG3**: see Figure 7) to avoid an unfair comparison. The think-aloud study provided us with a deeper insight into the behaviors of participants.

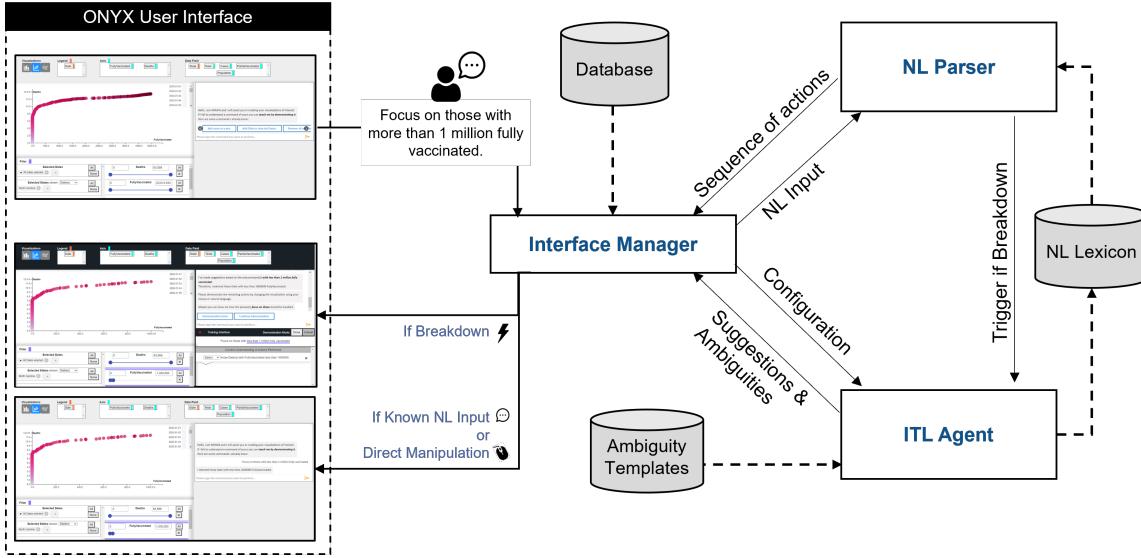


Figure 8: System Architecture Overview.

5.1 Participants

For our **online experiment**, we recruited 42 participants (20 male, 20 female, 2 non-binary) aged between 26 - 62 ($M = 40.3$, $SD = 8.8$) on Amazon Mechanical Turk. The participants were randomly assigned to one of the two conditions (baseline = 21; treatment = 21). Across both treatments, they exhibited minor experience in programming with 81% rating their experience as poor or fair and an average experience of 2.3 years in programming ($SD = 4.8$). 90% of our participants rated their experience with NLIs as either average or better than average. Participants are denoted as PO1.T-PO21.T for the treatment and PO1.B-PO21.B for the baseline condition in subsequent sections.

The 5 participants (3 male, 2 female) recruited for our **think-aloud evaluation** from a signup list of university students and the general public exhibited similar characteristics, with ages ranging between 23 - 68 ($M = 40.4$, $SD = 20.9$) and minor experience in programming (100% said their programming experience was poor). 3 out of the 5 participants (60%) rated their experience with NLIs as either average or better than average. All think-aloud participants were provided with the complete version of *ONYX*. These participants are denoted as PT1-PT5 in subsequent sections.

5.2 Procedure

At the beginning of both the online experiment and the think-aloud study, participants agreed to our IRB-approved consent form and were then provided with a pre-study questionnaire to elicit their demographics and their experience with programming and NLIs. Then, participants received an interactive guided tour that trained them in interacting with the data visualization tool. This tour did **not** include an introduction to the ITL aspects of *ONYX*. This provided participants with a basic understanding of the data visualization tool and ensured that the studies evaluated the effectiveness of the ITL and NLI aspects of *ONYX* and not the data

visualization aspects. The interactive guided tour took participants around 4 minutes to complete.

After the interactive guided tour, participants were provided with three tasks in a randomized order. Participants were required to proceed to the next task after they finished the demonstration process when they felt they accurately taught *ONYX* how to handle the NL input. After finishing they were not able to test their demonstration. The participants in the think-aloud study were additionally encouraged during this phase to think aloud and both their voice and the screen containing *ONYX* were recorded for later analysis.

After completing the tasks, participants were requested to fill out a post-study questionnaire regarding their subjective experience with the different features of *ONYX*. Finally, they were able to provide feedback about *ONYX* in a free-text field.

5.3 Tasks

The three tasks consisted of an NL input and a short description of its meaning, similar to previous evaluations of NLIs with ITL capabilities [23]. We derived the tasks based on common NL inputs in the formative study and ensured that key features of *ONYX* are covered by the tasks. To make sure that users needed to train the system with new NL inputs, we provided participants with NL inputs that *ONYX* did not yet know how to handle.

Task A: Demonstrating the NL input **Display all dates with more than 500**. This task focuses on providing suggestions (based on bold parts of the NL input) and addressing language ambiguities. Online participants in both treatments required on average 93 seconds (Min = 36, Max = 231, $SD = 46.8$) for the demonstration process, which was significantly faster than **Task B** and **Task C**.

Task B: Demonstrating the NL input **Only show Ohio and Florida on the 1st** which should *only* show Ohio and Florida for the 1st of September 2020 even if other states or dates had been previously displayed.

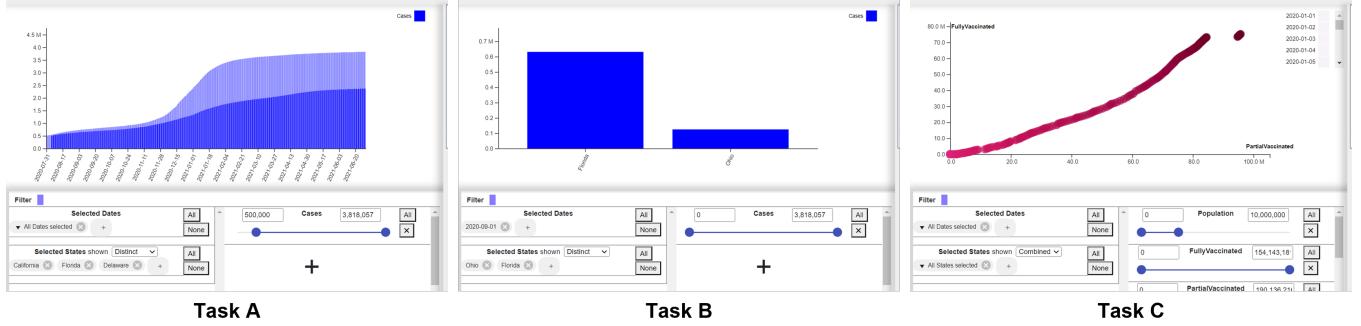


Figure 9: Target Visualizations after completing each Task A-C.

This task focuses on addressing direct manipulation ambiguities. Online participants in both treatments required on average 123 seconds (Min = 56, Max = 257, SD = 51) for the demonstration process.

Task C: Demonstrating the NL input Combine all states with less than
This task focuses on providing suggestions (based on bold parts of the NL input). Online participants in both treatments required on average 157 seconds (Min = 37, Max = 609, SD = 134.6) for the demonstration process.

We further confirmed that existing NLIs, such as Tableau's Ask Data and Microsofts Q&A, were not able to perform these NL inputs correctly without user involvement (e.g., in Task C, Ask Data only understood numeric filters and was *not* able to adapt the States filter accordingly or understood the goal of *combining*).

5.4 Results

5.4.1 Suggestions and Follow-Up Questions.

For participants in the online experiment, we measured the accuracy of how well their demonstration fit the meaning of the NL input by comparing the learned script from ONYX to a gold standard that was derived from the task descriptions. Specifically, for all three tasks, we calculated the accuracy as a percentage of how many of the requirements are included in the learned script and whether requirements are included that are not required based on the task description. Participants interacting with the full version of ONYX had significantly higher accuracy (median: 93.3%) than those in the baseline condition (median: 73.3%) ($U = 67.5$; $p < 0.001$), based on a Mann-Whitney U test (see Figure 10). The difference in the time it took the participants to demonstrate the NL inputs (averaged across all tasks) was not significant at a 0.05 level for the participants that interacted with the full version (median: 120 s) and the baseline version (median: 110.5 s) of ONYX ($U = 209$; $p = 0.78$).

For insights into why the accuracy of the two conditions of the online experiment differed, we analyzed the reason for the errors. We specifically investigated whether ONYX's follow-up questions and suggestions helped reduce the errors for the treatment condition in contrast to the baseline condition. Therefore, we labeled each incorrect section of the learned scripts based on whether ONYX provided no assistance for such errors (*other*) or if ONYX would assist in avoiding such errors through *follow-up questions* or *suggestions* (see Figure 11). In Task A and Task C, a third of the errors in the

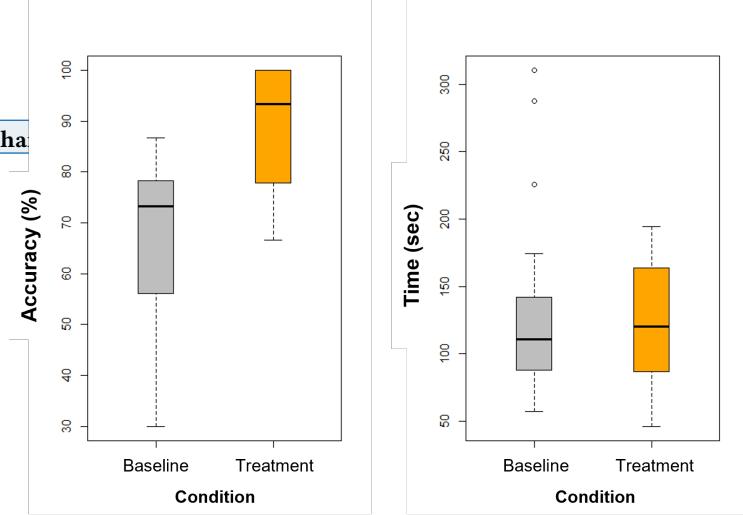


Figure 10: Boxplot of the Accuracy and Time for both Conditions for all 3 tasks. For Accuracy, a higher rating means better Accuracy. For Time, a lower rating means better Time.

baseline condition was associated with a lack of suggestions (Task A: 32.7%; Task C: 34.1%). A lack of follow-up questions targeting language ambiguities was associated with 18.8% of the errors in the baseline condition in Task A. In contrast, a lack of follow-up questions targeting direct manipulation ambiguities was associated with 84.2% of the errors in the baseline condition in Task B. However, participants in the treatment condition in Task B also exhibited errors associated with follow-up questions as they were able to ignore or decline the prompted follow-up questions.

Additionally, we asked participants in the post-study questionnaire of the treatment condition to rate statements regarding the effectiveness of suggestions and follow-up questions of ONYX on a 5-point Likert scale where 1 is "strongly disagree" and 5 is "strongly agree". ONYX achieved an average score of 4.3 on "*The assistant supports me through its suggestions*" and 4.3 on "*The assistant supports me through its follow-up questions to my actions*", further supporting the effectiveness of both the suggestions and follow-up questions (see Figure 12). These ratings were further supported by statements

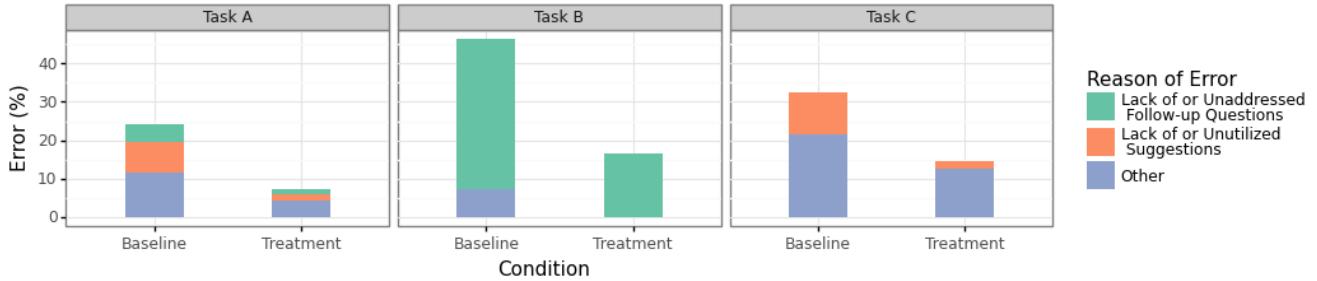


Figure 11: The Average Error in the Learned Scripts across Tasks A - C colored by the Reason of the Error.

made by participants and through a post-hoc analysis of the log data derived from user interactions.

Suggestions.

10 participants (47.6%) in the treatment condition of the online experiment explicitly stated that the suggestions helped them in their task in the free-text feedback field at the end of the post-study questionnaire. Specifically, through these suggestions participants ($n = 7$) understood what information ONYX required from them and helped them start demonstrating to ONYX how to handle the NL input. The statement “[ONYX] makes suggestions that are logical” by PO8.T further illustrates the statements from four participants that the suggestions they received were easy to interpret.

In the think-aloud study, we were able to investigate more closely how participants utilized the suggestions. All five participants noticed at the beginning of the demonstration mode what ONYX suggested, then checked if the suggestion fit their understanding of the NL input, and finally proceeded to demonstrate the part of the NL input ONYX did not yet know how to handle. For example, at the start of the demonstration PT4 said “*I would have done the same*” when checking the suggestions and PT3 said regarding the process of using the suggestions: “*I just had to check if it fits my own thinking*”. Two participants (40%) in the think-aloud study additionally stated that the suggestions helped them to stay in the analytic flow since the suggestions provided a good transition into the demonstration process without major disruptions.

Follow-up questions.

The positive aspects of the follow-up questions were explicitly mentioned by six participants (28.6%) in the treatment group of the online experiment in the free-text feedback field. Participants especially highlighted the understandability ($n = 4$) and timing ($n = 3$) of the follow-up questions. This is represented by the statement “*The questions were understandable and timely; that is, [ONYX] asked for clarifications at appropriate times and confirmed that it understood specific requests at appropriate times, as well*” by PO20.T. However, one participant in the online experiment also highlighted a negative aspect of the follow-up questions. PO15.T said: “*I found it difficult to demonstrate one NL input at a time and respond to the chat*”. Additionally, in the think-aloud study, two participants highlighted that they had problems with the follow-up questions at first because they did not notice the textual aid in the NLI as their focus was on another GUI element. For example, PT5 stated that the focus of their attention “*was over in the filter section. And not really looking at all the questions*”. This was supported by our log data as while

55.2% of follow-up questions were addressed by participants, 44.8% were unnoticed or incorrectly disregarded. However, PT5 further stated that it was easy to learn how to spot follow-up questions after noticing them for the first time, suggesting a learning effect.

To investigate this learning effect associated with follow-up questions closer, we analyzed the log data regarding the timing of follow-up questions prompted by ONYX and whether participants addressed them. Across all tasks, participants received a maximum of three follow-up questions per task. Participants sometimes received few or no follow-up questions during certain tasks which complicated the derivation of clear measures to assess the learning effect. Especially in Task C, only 4 participants received follow-up questions since performing the correct demonstrations did not trigger any follow-up questions. However, if participants still received a follow-up question due to additional incorrect demonstrations, they correctly addressed these follow-up questions. In Task A and Task B, all participants received at least one follow-up question. However, only six participants (28.6%) addressed this first follow-up question in Task A and only 10 (47.6%) in Task B. When participants received a second follow-up question, they increasingly addressed this follow-up question (Task A: 58.8%; Task B: 65%). In Task B, 11 participants received a third follow-up question, which was noticed and addressed by 81.8%.

5.4.2 Visual and Textual Aids.

Seven participants (16.7%) of both conditions in the online experiment highlighted in the free-text feedback that the visual and textual aids helped them better understand what ONYX understood and what ONYX did not understand in their NL input. This helped them “*learn what information [ONYX] needs*” as stated by PO19.T and two additional participants.

Based on the log data of participants, we analyzed for which concepts participants requested visual and textual aids. The aids can be categorized into visual and textual aids helping participants understand the concepts in the NL input (e.g., numeric relations, named entities) and visual aids helping participants understand follow-up questions (see Figure 14). While both conditions received visual and textual aids regarding the NL input, only the participants in the treatment condition could receive visual and textual aids regarding follow-up questions (see Section 4.2.4). If the NL input included concepts associated with named entities (e.g., Dates, Ohio, or 1st of September 2020), participants requested the connected visual and textual aid on average 1.82 times per task. Visual and

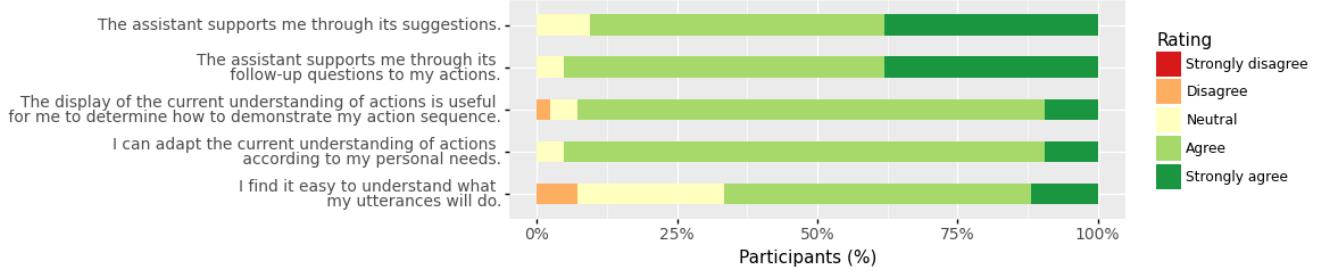


Figure 12: Responses to Post-Study Likert-scale Questions about the Experience of Participants with ONYX’s Features.

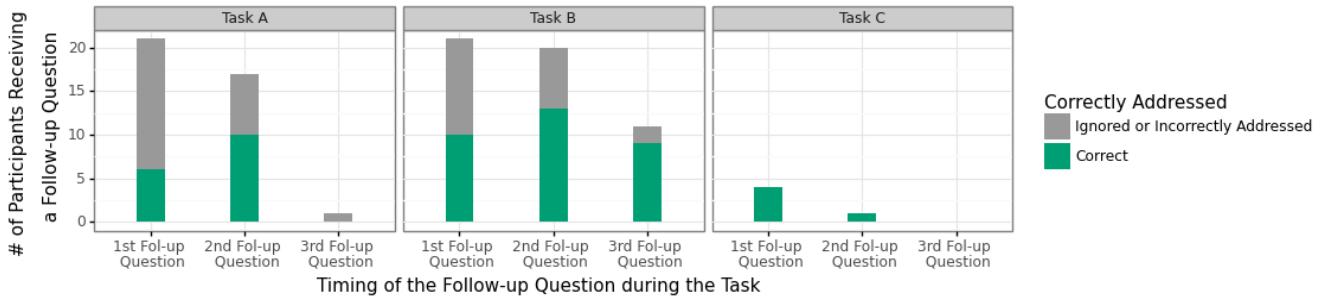


Figure 13: The Number of Participants receiving and addressing Follow-up Questions at different Timings during the Task

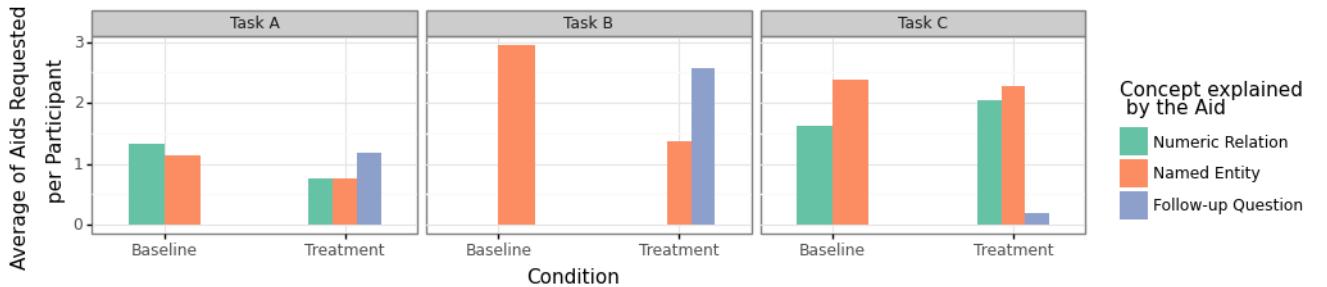


Figure 14: The Average Number of Visual and Textual Aids requested by Participants across Tasks A - C in the Baseline and Treatment Conditions. The Values are Colored by the Concept explained by the Visual and Textual Aid.

textual aids for numeric relations (e.g., more than 500,00 cases) were requested on average 1.44 times per task. Participants mostly requested visual aids for named entities and numeric relations *before* performing the first demonstration to get insights into the NLI’s understanding of their NL input. Participants requested visual aids for follow-up questions in the treatment condition on average 1.32 times per task and 0.29 times per follow-up question.

5.4.3 Display and Adaptation of ONYX’s Understanding.

Regarding the effectiveness of the visual representation of ONYX’s understanding during the demonstration, 93% of participants agreed or strongly agreed with “*The display of the current understanding of actions is useful for me to determine how to demonstrate my action sequence*”, 95% with “*I can adapt the current understanding of actions*

according to my personal needs” and 67% with “*I find it easy to understand what my utterances will do*” (see Figure 12).

6 DISCUSSION

The results of our formative study highlighted the need for assisting users in teaching NLIs how to handle new NL inputs through multi-modal ITL. The outcome of the user studies shows that ONYX’s suggestions, follow-up questions, and visual and textual aids are effective features to provide users with this kind of assistance. Participants in our user studies were able to significantly reduce errors in the learned scripts by requesting visual aids to receive a better understanding of ONYX’s knowledge, following ONYX’s suggestions, and clarifying the follow-up questions prompted by ONYX.

Assisting Users to Understand ONYX's Interpretations.

In our formative study, we learned that users had problems understanding how the earlier version of *ONYX* interpreted their NL input at the *start* of the demonstration process and how it interprets their demonstrations *during* the demonstration process. To address these issues, our final design of *ONYX* assists users through suggestions, follow-up questions, and visual and textual aids that are targeted at improving the users' understanding. Our quantitative and qualitative results of the final user studies provide evidence that *ONYX*'s clear suggestions at the start of the demonstration process assist users in understanding existing knowledge of *ONYX*. Through users' improved understanding, they are able to better derive what demonstrations they need to perform. We further learned that users are often unaware of possible ambiguities in their direct manipulation demonstrations. This is highlighted in Task B of the online experiment in which errors due to ambiguity accounted for 84.2% of the errors made by participants in the baseline condition. Providing assistance through follow-up questions helped *ONYX* reduce the errors due to ambiguities on average by 65.4%.

Learning to Utilize the Assistance.

In our user studies, we learned that users do not always follow the advice of the system in a mixed-initiative approach, even if the advice is correct. A central problem was that users first needed to learn *when* and *how* *ONYX* provided them with assistance. After they learned these two aspects they were able to improve their utilization of *ONYX*'s assistance in our user studies. Perhaps explicitly introducing users to the assistance the first time they receive it could further improve the utilization of *ONYX*'s assistance by users because their learning would be accelerated.

Making the Reason of Assistance Explicit.

Participants in our formative study highlighted that providing assistance without clarifying the reason for the assistance can even negatively influence the overall performance. This was because the assistance can disrupt the analytic flow. These findings highlight that more assistance is not always better. Our results on the final system suggest that users increasingly benefit from the assistance by *ONYX* if they are guided during the assistance through visual and textual aids. Furthermore, providing assistance directly after the event that triggered the need for assistance (synchronous assistance) helps users better map the assistance to its reason. Developers of ITL-based systems, therefore, need to ensure that the assistance is designed in such a manner that the benefit of the assistance outweighs its negative effects (e.g., interruptions). The certainty of the ITL-based system about the correctness of the assistance and its timing have to be taken into account to assess this trade-off.

Adaptability of *ONYX*'s Features.

Although we demonstrated *ONYX*'s capabilities in just one data visualization tool custom-built utilizing D3.js, we believe that the strategies *ONYX* uses would generalize to other data visualization tools. We point to similarities with other such tools, such as Power BI, Tableau, or open-source frameworks like vega-lite. For any of these, *ONYX*'s features could be adapted to assist users in teaching

the NLI how to specify (i) the encodings of the data visualization (e.g., the axes), (ii) the chart type, and (iii) numeric or categorical filters. *ONYX* is able to learn to perform these functionalities independent of the dataset when provided with the ability to track and manipulate the state of the data visualization tool either through an API or by directly integrating *ONYX* as in our case. However, *ONYX* is restricted to functionalities that users can perform through direct manipulation. Therefore, *ONYX* is, for example, not able to learn to internally calculate additional metrics, such as the average across the visualized data points, unless there is a built-in control for that in the GUI. We further believe that with additional future work our insights can be adapted to domains in which systems possess clear pre-defined functionalities and the state of the system can be translated into structured data. These insights might then be applied to augment existing ITL-based systems, such as VoiceCuts integrated into Photoshop [17].

7 LIMITATIONS AND FUTURE WORK

NL Inputs with Multiple Meanings.

ONYX cannot distinguish between multiple meanings of the same NL input based on the situational context. For example, the NL input **(Remove states)** was used by users in the formative study both (i) to remove states from the x-Axis and (ii) to remove the filter associated with states. Therefore, the situational context, such as current configuration, previous interactions, or time and location of users, would be informative and relevant for *ONYX* to distinguish between similar NL inputs with varying meanings in different situational contexts. An important focus of our future work is to enable users to clarify the aspects of the situational context that lead to different meanings of similar NL inputs and to enable *ONYX* to learn from these clarifications.

Recognition of Important Words, Synonyms, and Antonyms.

The current NL Parser enables *ONYX* to distinguish between NL inputs based on task-related keywords, the parameters, and the sentence structure. However, *ONYX* has no semantic understanding of important words contained in the NL input. For example, when *ONYX* learns how to interpret **(Only show Florida)**, it does *not* learn what the specific word **Only** means. Furthermore, the NL Parser is not able to identify synonyms and antonyms. To address these shortcomings, we plan to integrate post-processing of the learned NL inputs utilizing TF-IDF and the dependency structure to identify important words and their connection with parameters and parts of the scripts. This could then be utilized to inform additional suggestions provided by *ONYX* to assist users during the demonstration process. Furthermore, we plan to utilize open source knowledge provided by ontologies (e.g., WordNet, VerbNet) to inform this post-processing and to enable *ONYX* to process synonyms and antonyms.

Cold-Start Problem.

Lastly, *ONYX* initially only knew how to handle a limited set of NL inputs without additional training and was therefore only able to support users with suggestions based on a limited set of user-defined procedures. To address this issue, we plan to integrate our

ITL approach with *existing* NL toolkits for creating data visualizations (e.g., NL4DV [33], which is grammar-based, or ncNet [26], which is machine learning-based) by extracting existing knowledge from these NL toolkits and providing this knowledge through *ONYX* to users throughout the demonstration process. Through this approach, the ITL capabilities of *ONYX* would not be an alternative, but an extension to the existing advances in NL processing, such as systems similar to GPT-3. This would be beneficial in that while insights from machine learning-based NLIs can be more easily scaled across different situations, existing NLIs based on machine learning still require numerous examples to train new tasks and are a black box to users (e.g., [26]).

8 CONCLUSION

Users are increasingly empowered to personalize natural language interfaces (NLIs) by teaching how to handle new natural language (NL) inputs. In this paper, we introduce *ONYX* which integrates a multi-modal interactive task learning (ITL) approach that assists users during the demonstration process to improve the accuracy of the learned script. Specifically, *ONYX* assists users through suggestions based on parts of the NL input *ONYX* understood, follow-up questions to address ambiguities in direct manipulation demonstrations, and guidance through visual and textual aids. The results of our user studies show that the proposed *ONYX* features help users significantly improve the accuracy of the learned script for the NL input without requiring more time. Furthermore, participants appreciated how these features are integrated into *ONYX* and how we addressed the features' trade-offs. More broadly, our work demonstrates how users can be assisted during the demonstration process by an ITL agent to create a synergistic experience in personalizing an NLI in a multi-modal system.

REFERENCES

- [1] 2019. Create commands to control online services and devices. <https://support.google.com/googlenest/answer/7194656>
- [2] 2022. Shortcuts and Suggestions - Siri - Human Interface Guidelines - Apple Developer. <https://developer.apple.com/design/human-interface-guidelines/siri/overview/shortcuts-and-suggestions/>
- [3] James Allen, Nathanael Chambers, George Ferguson, Lucian Galescu, Hyuckchul Jung, Mary Swift, and William Tayson. 2007. PLOW: A collaborative task learning agent. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 2. AAAI Press, 1514–1519. <https://doi.org/10.5555/1619797.1619888>
- [4] 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*. ACM, New York, NY, USA, 1–12. <https://doi.org/10.1145/3290605.3300484>
- [5] Amos Azaria, Shashank Srivastava, Jayant Krishnamurthy, Igor Labutov, and Tom M. Mitchell. 2020. An agent for learning new natural language commands. *Autonomous Agents and Multi-Agent Systems* 34, 1 (2020), 6. <https://doi.org/10.1007/s10458-019-09425-x>
- [6] Tracey Booth and Simone Stumpf. 2013. End-user experiences of visual and textual programming environments for Arduino. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 7897 LNCS. 25–39. https://doi.org/10.1007/978-3-642-38706-7_4
- [7] M. Bostock, V. Ogievetsky, and J. Heer. 2011. D³: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (12 2011), 2301–2309. <https://doi.org/10.1109/TVCG.2011.185>
- [8] Julia Cambre, Alex C. Williams, Afsaneh Razi, Ian Bicking, Abraham Wallin, Janice Tsai, Chimmay Kulkarni, and Jofish Kaye. 2021. Firefox Voice: An Open and Extensible Voice Assistant Built Upon the Web. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–18. <https://doi.org/10.1145/3411764.3445409>
- [9] Xin Chen, Jessica Zeitz Self, Leanna House, John Wenskovitch, Maoyuan Sun, Nathan Wycoff, Jane Robertson Evia, Scotland Leman, and Chris North. 2018. Be the Data: Embodied Visual Analytics. *IEEE Transactions on Learning Technologies* 11, 1 (2018), 81–95. <https://doi.org/10.1109/TLT.2017.2757481>
- [10] Ensheng Dong, Hongru Du, and Lauren Gardner. 2020. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases* 20, 5 (5 2020), 533–534. [https://doi.org/10.1016/S1473-3099\(20\)30120-1](https://doi.org/10.1016/S1473-3099(20)30120-1)
- [11] James R. Eagan and John T. Stasko. 2008. The Buzz: Supporting user tailorability in awareness applications. *Conference on Human Factors in Computing Systems - Proceedings* (2008), 1729–1738. <https://doi.org/10.1145/1357054.1357324>
- [12] Michael H Fischer, Giovanna Campagna, Euirim Choi, and Monica S Lam. 2021. DIY assistant: A multi-modal end-user programmable virtual assistant. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. ACM, 312–327. <https://doi.org/10.1145/3453483.3454046>
- [13] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie Karahalios. 2015. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*. ACM Press, New York, New York, USA, 489–500. <https://doi.org/10.1145/2807442.2807478>
- [14] Jonathan Grudin and Richard Jacques. 2019. Chatbots, Humbots, and the Quest for Artificial General Intelligence. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, Vol. 11. ACM, New York, NY, USA, 1–11. <https://doi.org/10.1145/3290605.3300439>
- [15] Piedade João, Dorotea Nuno, Sampaio Ferrentini Fábio, and Pedro Ana. 2019. A cross-analysis of block-based and visual programming apps with computer science student-teachers. *Education Sciences* 9, 3 (2019). <https://doi.org/10.3390/educsci9030181>
- [16] Young-Ho Kim, Bongshin Lee, Arjun Srinivasan, and Eun Kyoung Choe. 2021. Data@Hand: Fostering Visual Exploration of Personal Data on Smartphones Leveraging Speech and Touch Interaction. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–17. <https://doi.org/10.1145/3411764.3445421>
- [17] Yea-Seul Kim, Mira Dontcheva, Eytan Adar, and Jessica Hullman. 2019. Vocal Shortcuts for Creative Experts. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, Vol. 14. ACM, New York, NY, USA, 1–14. <https://doi.org/10.1145/3290605.3300562>
- [18] Andrew J. Ko, Brad A. Myers, Michael J. Coblenz, and Htet Htet Aung. 2006. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Transactions on Software Engineering* 32, 12 (12 2006), 971–987. <https://doi.org/10.1109/TSE.2006.116>
- [19] Rebecca Krosnick and Steve Oney. 2022. ParamMacros : Creating UI Automation Leveraging End-User Natural Language Parameterization. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*.
- [20] John E Laird, Kevin Gluck, John Anderson, Kenneth D Forbus, Odest Chadwicke Jenkins, Christian Lebiere, Dario Salvucci, Matthias Scheutz, Andrea Thomaz, Greg Trafton, Robert E Wray, Shiwali Mohan, and James R Kirk. 2017. Interactive Task Learning. *IEEE Intelligent Systems* 32, 4 (2017), 6–21. <https://doi.org/10.1109/MIS.2017.3121552>
- [21] Gilly Leshed, Eben M Haber, Tara Matthews, and Tessa Lau. 2008. CoScripter. In *Proceeding of the 26th annual CHI conference on Human factors in computing systems - CHI '08*. ACM Press, New York, New York, USA, 1719. <https://doi.org/10.1145/1357054.1357323>
- [22] Chi-Hsun Li, Su-Fang Yeh, Tang-Jie Chang, Meng-Hsuan Tsai, Ken Chen, and Yung-Ju Chang. 2020. A Conversation Analysis of Non-Progress and Coping Strategies with a Banking Task-Oriented Chatbot. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–12. <https://doi.org/10.1145/3313831.3376209>
- [23] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, Vol. 2017-May. ACM, New York, NY, USA, 6038–6049. <https://doi.org/10.1145/3025453.3025483>
- [24] Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Wanling Ding, Tom M. Mitchell, and Brad A. Myers. 2018. APPINITE: A Multi-Modal Interface for Specifying Data Descriptions in Programming by Demonstration Using Natural Language Instructions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Vol. 2018-Octob. IEEE, 105–114. <https://doi.org/10.1109/VLHCC.2018.8506506>
- [25] 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*. ACM, New York, NY, USA, 577–589. <https://doi.org/10.1145/3332165.3347899>
- [26] Yuyu Luo, Nan Tang, Guoliang Li, Jiawei Tang, Chengliang Chai, and Xuedi Qin. 2022. Natural Language to Visualization by Neural Machine Translation. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (1 2022), 217–226. <https://doi.org/10.1109/TVCG.2021.3114848>
- [27] Rodrigo De A. Maués and Simone Diniz Junqueira Barbosa. 2013. Keep doing what I just did: Automating smartphones by demonstration. *MobileHCI 2013 - Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services* (2013), 295–303. <https://doi.org/10.1145/2493190.2493216>

- [28] Microsoft. 2022. Teach Q&A to understand questions and terms in Power BI Q&A - Power BI | Microsoft Docs. <https://docs.microsoft.com/en-us/power-bi/natural-language/q-and-a-tooling-teach-q-and-a>
- [29] Brad A. Myers, Amy J. Ko, Thomas D. LaToza, and Youngseok Yoon. 2016. Programmers Are Users Too: Human-Centered Methods for Improving Programming Tools. *Computer* 49, 7 (7 2016), 44–52. <https://doi.org/10.1109/MC.2016.200>
- [30] Brad A. Myers, Andrew J. Ko, Chris Scaffidi, Stephen Oney, Young Seok Yoon, Kerry Chang, Mary Beth Kery, and Toby Jia Jun Li. 2017. Making end user development more natural. In *New Perspectives in End-User Development*. 1–22. https://doi.org/10.1007/978-3-319-60291-2_1
- [31] Brad A Myers, Richard G. McDaniel, and David S Kosbie. 1993. Marquise. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '93*. ACM Press, New York, New York, USA, 293–300. <https://doi.org/10.1145/169059.169225>
- [32] Arpit Narechania, Adam Fourney, Bongshin Lee, and Gonzalo Ramos. 2021. DIY: Assessing the Correctness of Natural Language to SQL Systems. In *26th International Conference on Intelligent User Interfaces*. ACM, New York, NY, USA, 597–607. <https://doi.org/10.1145/3397481.3450667>
- [33] Arpit Narechania, Arjun Srinivasan, and John Stasko. 2021. NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2 2021), 369–379. <https://doi.org/10.1109/TVCG.2020.3030378>
- [34] Saziyi Betül Özateş, Arzucan Özgür, and Gomir R. Draradev. 2016. Sentence similarity based on dependency tree kernels for multi-document summarization. In *Proceedings of the 10th International Conference on Language Resources and Evaluation, LREC 2016*. 2833–2838. <http://duc.nist.gov/duc2003/tasks.html>
- [35] Lihang Pan, Chun Yu, JiaHui Li, Tian Huang, Xiaojun Bi, and Yuanchun Shi. 2022. Automatically Generating and Improving Voice Command Interface from Operation Sequences on Smartphones. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–21. <https://doi.org/10.1145/3491102.3517459>
- [36] Alborz Rezazadeh Sereshkeh, Gary Leung, Krish Perumal, Caleb Phillips, Minfan Zhang, Afsaneh Fazly, and Iqbal Mohamed. 2020. VASTA. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, Vol. 20. ACM, New York, NY, USA, 22–32. <https://doi.org/10.1145/3377325.3377515>
- [37] 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 (UIST '16)*. ACM, New York, NY, USA, 365–377. <https://doi.org/10.1145/2984511.2984588>
- [38] Arjun Srinivasan, Bongshin Lee, Nathalie Henry Riche, Steven M. Drucker, and Ken Hinckley. 2020. InChorus: Designing Consistent Multimodal Interactions for Data Visualization on Tablet Devices. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–13. <https://doi.org/10.1145/3313831.3376782>
- [39] Arjun Srinivasan, Nikhil Nyapathy, Bongshin Lee, Steven M. Drucker, and John Stasko. 2021. Collecting and Characterizing Natural Language Utterances for Specifying Data Visualizations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–10. <https://doi.org/10.1145/3411764.3445400>
- [40] Gavin Sudrey, Ben Talbot, and Frederic Maire. 2022. Learning and Executing Re-Usable Behaviour Trees From Natural Language Instruction. *IEEE Robotics and Automation Letters* 7, 4 (10 2022), 10643–10650. <https://doi.org/10.1109/LRA.2022.3194681>
- [41] Tableau. 2019. Optimize Data for Ask Data - Tableau. <https://help.tableau.com/current/pro/desktop/en-us/askdataoptimize.htm>
- [42] Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. 2015. Learning to interpret natural language commands through human-robot dialog. *IJCAI International Joint Conference on Artificial Intelligence 2015-Janua*, Ijcai (2015), 1923–1929.
- [43] Melanie Tory and Vidya Setlur. 2019. Do What I Mean, Not What I Say! Design Considerations for Supporting Intent and Context in Analytical Conversation. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 93–103. <https://doi.org/10.1109/VAST47406.2019.8986918>
- [44] Priyan Vaithilingam and Philip J. Guo. 2019. Bespoke: Interactively synthesizing custom GUIs from command-line applications by demonstration. In *UIST 2019 - Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 563–576. <https://doi.org/10.1145/3332165.3347944>
- [45] Sida I. Wang, Samuel Ginn, Percy Liang, and Christopher D. Manning. 2017. Naturalizing a Programming Language via Interactive Learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vol. 1. Association for Computational Linguistics, Stroudsburg, PA, USA, 929–938. <https://doi.org/10.18653/v1/P17-1086>
- [46] Jonathan Zong, Dhiraj Barnwal, Rupayan Neogy, and Arvind Satyanarayanan. 2021. Lyra 2: Designing Interactive Visualizations by Demonstration. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2 2021), 304–314. <https://doi.org/10.1109/TVCG.2020.3030367>