# Follow the Successful Herd: Towards Explanations for Improved Use and Mental Models of Natural Language Systems

Michelle Brachman
michelle.brachman@ibm.com
IBM Research
Cambridge, MA, USA

Qian Pan
qian.pan@ibm.com
IBM Research
Cambridge, MA, USA

Hyo Jin Do
hjdo@ibm.com
IBM Research
Cambridge, MA, USA

Casey Dugan
cadugan@us.ibm.com
IBM Research
Cambridge, MA, USA

Arunima Chaudhary
aruc540@gmail.com
LinkedIn
Bengaluru, Karnataka, India

James M. Johnson
jmjohnson@us.ibm.com
IBM Research
Cambridge, MA, USA

Priyanshu Rai
priyanshu.rai@ibm.com
IBM Research
Pune, MH, India

Tathagata Chakraborti
tathagata.chakraborti1@ibm.com
IBM Research
Cambridge, MA, USA

Thomas Gschwind
THG@zurich.ibm.com
IBM Research
Zurich, Switzerland

Jim Laredo
laredoj@us.ibm.com
IBM Research
Yorktown Heights, NY, USA

Christoph Miksovic
CMI@zurich.ibm.com
IBM Research
Zurich, Switzerland

Paolo Scotton
psc@zurich.ibm.com
IBM Research
Zurich, Switzerland

Kartik Talamadupula
krtalamad@us.ibm.com
IBM Research
Seattle, WA, USA

Gegi Thomas
gegi@us.ibm.com
IBM Research
Yorktown Heights, NY, USA

## ABSTRACT

While natural language systems continue improving, they are still imperfect. If a user has a better understanding of how a system works, they may be able to better accomplish their goals even in imperfect systems. We explored whether explanations can support effective authoring of natural language utterances and how those explanations impact users' mental models in the context of a natural language system that generates small programs. Through an online study (n=252), we compared two main types of explanations: 1) system-focused, which provide information about how the system processes utterances and matches terms to a knowledge base, and 2) social, which provide information about how other users have successfully interacted with the system. Our results indicate that providing social suggestions of terms to add to an utterance helped users to repair and generate correct flows more than system-focused explanations or social recommendations of words to modify. We also found that participants commonly understood some mechanisms of the natural language system, such as the matching of terms to a knowledge base, but they often lacked other critical knowledge, such as how the system handled structuring and ordering. Based on these findings, we make design recommendations for supporting interactions with and understanding of natural language systems.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs International 4.0 License.

*IUI '23, March 27–31, 2023, Sydney, NSW, Australia*
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0106-1/23/03.
https://doi.org/10.1145/3581641.3584088

## CCS CONCEPTS

• **Human-centered computing → Natural language interfaces**; *Empirical studies in HCI*;

## KEYWORDS

natural language interaction, AI explainability, mental models

**ACM Reference Format:**
Michelle Brachman, Qian Pan, Hyo Jin Do, Casey Dugan, Arunima Chaudhary, James M. Johnson, Priyanshu Rai, Tathagata Chakraborti, Thomas Gschwind, Jim Laredo, Christoph Miksovic, Paolo Scotton, Kartik Talamadupula, and Gegi Thomas. 2023. Follow the Successful Herd: Towards Explanations for Improved Use and Mental Models of Natural Language Systems. In *28th International Conference on Intelligent User Interfaces (IUI '23), March 27–31, 2023, Sydney, NSW, Australia.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3581641.3584088

## 1 INTRODUCTION

Researchers have advocated for the use of natural language interfaces for decades [53, 134]. Recently, advances in natural language processing (NLP) have made natural language interfaces possible for a variety of tasks, such as interacting with smart home devices[1], analyzing data [32, 39, 43, 54, 110, 138], and programming [51, 69, 70, 72, 94, 98, 121, 137], among others. In complex contexts, natural language interfaces may reduce complexity, enabling non-experts to use previously inaccessible systems and lowering effort for experts [79, 104]. Additionally, generative AI systems are becoming more prevalent, such as systems that generate artifacts like code, text, or images, often from natural language [68, 108, 113]. In this work, we consider a system in which users can enter a natural language *utterance*, which the system processes to create a *flow*, or a small trigger-action program.

While natural language technologies have come a long way, users of natural language systems still often find themselves misunderstood. One reason for this is that natural language systems are still imperfect [48, 61, 95]. Lack of understanding of a system can lead to gulfs between users' expectations of systems and the reality of what the systems can do [57–59, 74, 83]. This leads to failures in the use of natural language systems, resulting in time lost and frustration for users [16, 60, 139]. Researchers have begun to consider how to help users repair during breakdowns, at which point users need to figure out how to modify their input to receive their desired output [7, 71, 131].

Explanations may be one way to help users better understand and work with intelligent systems, including NLP systems. We use a recent definition of explanations: 'any means of helping users achieve a better understanding of the AI system' [115]. Explanations have typically been studied in the context of classification tasks or decision systems [3], often considering how explanations impact users' trust in those systems [100, 119]. However, research on explanations in systems that generate artifacts has only just begun, showing that users have unique needs in these contexts [115]. Gaps also exist in our understanding of how people use and understand explanations for natural language interfaces [30]. Our work contributes to the understanding of how people use explanations to interact with and understand a natural language system that generates flows.

In this work, we designed and implemented six explanation types based on existing work in explanations [36, 115, 132]. We aimed to provide insight into how our system functions, either through 1) system-focused explanations, which provide information about how the flow is created based on the natural language, or 2) social explanations, which provide information about how others successfully used the system. Our system-focused explanations represent the many explanations which focus on providing information about how a system processes inputs and generates output. In contrast, our social explanations provide insight into how other users modified their inputs to move toward more successful task outcomes. While these types of explanations have been suggested and included in systems, little work has evaluated their impact on mental models and users' abilities to repair unsuccessful interactions with a natural language system.

To explore our explanations for a natural language to flow system, we ran a between-subjects study on Amazon Mechanical Turk (n=252). Users had five attempts to create flows using the natural language system, with one of our explanation types or without an explanation. We had three research questions:

- RQ1: How did explanations impact repair success, overall correctness, and efficiency?
- RQ2: How did participants use explanations?
- RQ3: How did explanations impact users' mental models of the system?

Our social explanations, which suggested words or phrases to add to an utterance, helped users more than system explanations or explanations that showed words to modify. Participants also modified their utterances more often using social explanations than system explanations. However, we did not find differences in users' mental models across explanation types. While many participants understood that the system was attempting to match their words to a known set, many participants did not seem to have a full understanding of other aspects of the system, like structuring and ordering. Our contributions are: 1) an empirical comparison of explanation styles for a natural language system, 2) insight into users' mental models of a natural language system that generates flows, and 3) design recommendations for natural language systems.

## 2 RELATED WORK

Our work contributes to efforts in natural language interface support, explainable AI, and mental models of AI systems.

### 2.1 Natural Language Interface Support

Our work broadly fits into the area of support for natural language interfaces, and in particular, providing support to users inputting natural language utterances and repairing their natural language inputs in the case of a breakdown.

*2.1.1 Input support.* Natural language interfaces commonly provide one or more ways to support users inputting their utterances. For example, systems provide auto-complete [17, 32, 54, 110, 121, 138], suggested utterances or examples [29, 32, 42, 114], multiple alternate outputs [59, 102], and direct manipulation methods [43, 54, 69, 110]. Yet, only a few studies have considered the effectiveness of particular types of input support for users. Comparisons of natural language systems have shown benefits of widgets and auto-complete [43, 110]. Providing an explanation and options for disambiguation also helped users [69]. However, variations on suggested commands impacted the number of commands but not users' success rate [114]. Our study expands on knowledge about input support for natural language systems through a comparative study of explanations.

*2.1.2 Failure handling in natural language processing.* Due to the challenges in interacting with natural language interfaces, researchers have explored ways to support the 'repair' of breakdowns in natural language systems. Breakdowns occur when a system does not understand the user's input, and repair is the process of "recovering from the breakdown and accomplishing the task goal" [7]. To perform repairs, users often attempt to expand or reduce the scope of their input [59], or reword their input [59, 133], but still need more support to be successful. Researchers have found that agents should acknowledge breakdowns and suggest ways to

[1]Amazon Alexa (https://alexa.amazon.com), Apple Siri (https://www.apple.com/siri/), Google Nest (https://store.google.com/category/connected_home), etc

repair [7], such as by correcting a previous input, or through ambiguity widgets [54]. Other systems have helped users repair through instructions, interactions and examples [131], as well as using direct manipulation through widgets [71]. We contribute new knowledge on how explanations impact repair in a natural language system.

## 2.2 Explainable AI

There is a long history of research addressing the interpretability and explainability of machine learning and AI [1, 6, 46, 49, 78, 119]. Interpretable machine learning is "the science of comprehending what a model did (or might have done)" [46], while explainable AI (XAI) "aims to make AI system results more understandable to humans" [1]. We situate our work in the context of human-centered XAI, explanations for natural language and generative AI systems, explanations for automated planning, and social explanations.

*2.2.1 Human-Centered Explainable AI.* Explanations may have a variety of benefits to users, such as enabling users to assess trust and decide between models [100] and reducing bias [106]. However, explanations do not always have the intended impacts [129] and can cause misuse of AI systems [12, 35, 38, 75, 136]. Further, explanations can be individual to particular systems [75] and users [117]. Researchers have developed frameworks and question banks to support design of explanations [37, 73, 80, 127]. However, much of the work in the design and evaluation of human-centric XAI focuses on decision-based systems, such as medical decision systems [56, 96, 120, 135], bail decision systems [4, 33, 77], and recommendation systems [62, 86, 93]. These explanations often focus on feature relevance [19, 52, 129] and the impact of explanations on trust [100, 136], persuasiveness [62], and acceptance of systems [50]. Our work addresses how explanations can support user performance working with systems. The impact of explanations on task performance has been mixed: explanations improved success on a recall quiz for game events [101], but not for human-AI task assignment [105] or collaborative performance for decision-making [12]. Further, the evaluation of explanations with proxy tasks and subjective measures may not align with true performance [21]. The work most related to ours is a study of three types of explanations for supporting human task performance in crowd ideation [130]. In this study, users understood the types of explanations similarly and explanations that highlighted relevant words were the most useful, compared to those that provided counterfactual suggestions. We consider a set of similar, but different explanation types in the context of a natural language system for generating flows and evaluate performance as well as users' mental models.

*2.2.2 Explainable AI for Natural Language & Generative Systems.* We designed and evaluated explanations in a natural language system that outputs small trigger-action flows, similar to generative systems. Researchers have investigated explanations for these two particular contexts, in which users may have specific needs.

Significant work in machine learning communities has addressed explainability, interpretability, and transparency in natural language processing (NLP) [30, 84, 99]. At a high level, explanations for NLP systems are either local (for a specific input) or global (for a model as a whole) and either self-explaining (explaining a model using the model itself) or post-hoc (additional operations after a model has made a prediction) [30]. The most common types of explanations are local using feature importance, such as highlighting input text or showing relations [30, 84]. Other explainability techniques include using a surrogate model, showing examples, showing the process by which a prediction was derived, or natural language examples. While these explanations are often evaluated in automated ways, researchers argue for human evaluation of explanations [84]. Some research with humans has shown positive, but nuanced results, such as that some types of highlighting may be more helpful than others [90].

Researchers have also developed methods for explaining the outputs of generative AI models, in which users have different needs than for discriminative machine learning [81, 103, 115]. For example, sliders and multiple alternative outputs improved users' attitudes toward the AI and supported new interaction methods [81, 82]. Confidence highlighting and alternate outputs supported users in understanding a generative model [132]. Finally, an interactive method for evaluating generated images resulted in high quality and diverse images [140]. We build upon research in explainability and support for users of generative models through the study of how explanation types impact users' ability to successfully generate small trigger-action programs from natural language. We leverage explanation types from prior work in the design of our explanations (see Section 4).

*2.2.3 Explainable AI for Automated Planning.* The final AI component of our system – an automated planner – processes the information derived from the natural language interface in the back-end. This component also comes with a long history of research [5, 23, 24, 41, 66] in Explainable AI Planning (XAI/P) primarily focused on two threads: explaining the logical constraints and rules that support a flow [109] and explaining the knowledge that supports those constraints [25] in terms of differences between the user and the system models, the latter among the first algorithms from the XAI/P community to support the contrastive, social, and selective properties of explanations [87]. While a comprehensive explanation of how our system works end-to-end would likely involve explanations from both components – the natural language processing unit in the front-end as well as the automated planner that drives the back-end – in this paper, we focus only on the explanations of how the natural language user inputs are processed. How the individual components in the flow were constructed is perhaps more relevant for developers, power users, and subject matter experts.

*2.2.4 Social Explanations.* A recent push in XAI research has been towards explanations that are socially-situated [36, 75, 115] and based on knowledge from the social sciences [87]. Social explanations have been primarily explored in recommendation systems [26, 34, 97, 111] and AI-supported decision making contexts [36]. They improved users' perceptions of the quality of recommendations [97], increased the likelihood of users considering a recommendation [111], and supported the calibration of trust [36]. Our work builds upon prior work through the evaluation of three types of social explanations for a natural language system that generates flows.

### 2.3 Mental Models of AI Systems

One way to evaluate whether an explanation is supporting users in understanding a system is to evaluate users' mental models. Theories of mental models exist broadly, such as in human-computer interaction, which considers how people think about systems [91], as well as other fields like education [47]. Our work is particularly related to the types and correctness of mental models, how mental models change over time, and ways to improve mental models of AI systems.

Users often have differing mental models of the same system and varying degrees of correctness. One study in a game context showed that users' mental models were almost always accurate, though none of the participants were able to win the game [133]. Other studies have shown more variation in users' mental modes, such as for virtual personal assistants [118] and robots [10]. Mental models may also be impacted by social factors, such as what a user thinks a human would know [67], or media representations [10, 126]. Several studies have considered the connection between soundness of mental models and success interacting with a system, such as in a game setting [11, 45] and when using a music recommender system [64]. Further, mental models likely aren't static, as users iterate on their mental models through exploration and elaboration [27, 125] and learn about systems over time [14, 15]. The evolution of mental models also relates to the 'Theory of Mind,' which considers how people think about others' minds based on interactions [13, 122]. These studies show that people's perceptions and beliefs change and evolve over time, such as based on the physical environment in a Minecraft-like context [13], and through conversation with a conversational agent [128]. Some methods may support the improvement of mental models, such as tutorials [64] and explanations [65], though results for explanations have been somewhat mixed. Explanations for a task assignment process improved users' mental models [105], but non-technical users may overestimate the correctness of their mental models [28]. Mental model accuracy can also be affected by a variety of contextual factors, like type of explanation, expertise [107, 117, 129], personal characteristics [86, 93], and order of seeing system strengths and weaknesses [92]. We contribute to the understanding of users' mental models of AI systems through users' agreement with statements about our system and users' descriptions of how the system works.

## 3 CONTEXT: NATURAL LANGUAGE SYSTEM FOR GENERATING TRIGGER-ACTION FLOWS

We explore the impact of explanation types within a natural language interface that generates trigger-action flows. We describe flows, user interaction in our system, and finally how the system processes users' natural language.

### 3.1 Application Integration Automation Flow

We use the term 'flow' to describe a short trigger-action program designed to automate application integration. In our context, the first component in a flow is the trigger, or the event that causes the following actions to take place. Each component has three properties: the application name, the operation, and the object. For example, a component could have the application *Gmail*, the operation *create* and the object *email*, which would create a new email in Gmail. Our flows are similar to 'if this, then that (IFTTT)' programs, commonly used in the internet of things or for consumer phone automation [2]. In business contexts, flows can automate application integration. For example, imagine a marketing specialist who finds customer leads. When they enter a new lead into an application like Salesforce, they might want to also add the user to their email list in Gmail and send a Slack message to their team about the new lead. Various tools support the creation of this type of flow, such as Workato[3], IBM App Connect[4], and SnapLogic[5]. Systems often provide code-based and GUI-based methods for creating these flows, such that users with a wide range of technical knowledge can create them. In our system, users can generate a flow by writing an utterance in natural language that describes their flow. Entering the flow description using an utterance has two main advantages over the use of the GUI: (1) the user does not have to navigate multiple menus that they might not be familiar with, and (2) the processing of the utterance acts as a sophisticated search function, finding appropriate objects through their description in natural language and generating a flow.

### 3.2 User Interaction

The user interacts with our system by entering an utterance, which is a one-sentence description of their flow in English, as shown in Figure 1. The system processes the natural language and then displays the generated flow for the user. The version of the system evaluated in this paper produces a flow template without hooking into a user's specific accounts and data. The current system also requires that the user input natural language with common English words which are translated to technical terms within the system, such as 'message' in the input and 'rawmessage' in the generated flow in Figure 1. Flows are displayed visually as a series of components, with the application, object, and operation displayed on them. The system by default does not provide any information about how the natural language processing or flow generation occurs other than the generated flow. The interface provides one text example directly above the textbox to help a user understand the type of utterance they can provide.

### 3.3 Natural language to flow implementation

The system has three steps, which together form a pipeline for generating the flow: parsing, matching, and ordering (full technical description is available in Appendix A). Each step passes the processed information to the next step, eventually producing one generated flow. The way the system performs these processes impacts how a user needs to write their utterance and what they might need to change as part of repair in the case of a breakdown. We used the following conceptual model of how flows are generated to evaluate users' mental models (see section 5.1.2). As noted in Section 2.2.3, the explanations only surface combined details from

[2]https://ifttt.com/
[3]https://www.workato.com/
[4]https://www.ibm.com/cloud/app-connect
[5]https://www.snaplogic.com/

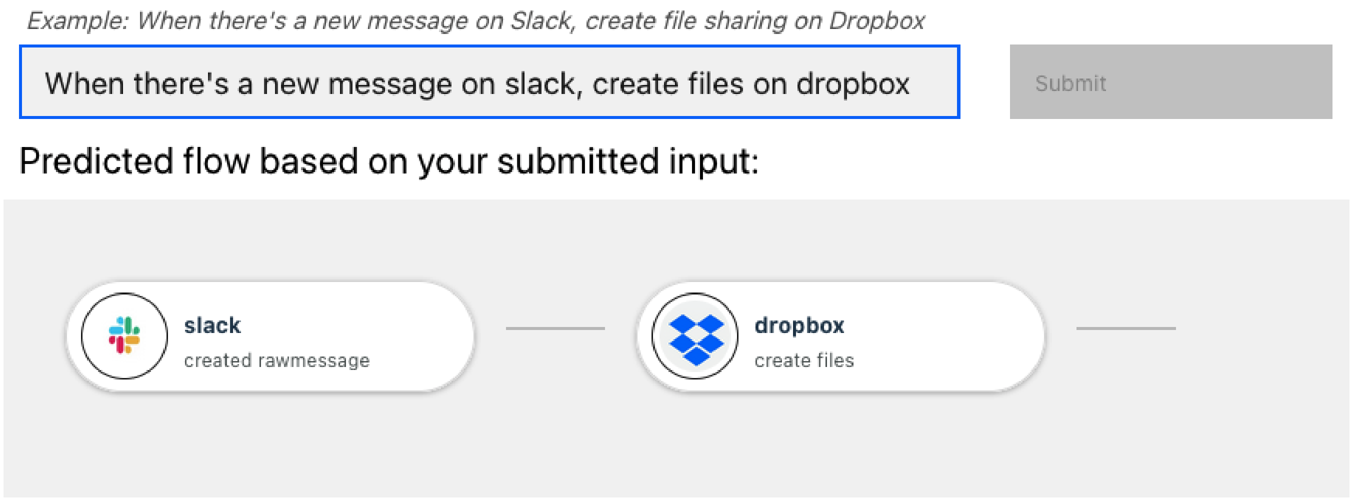

Example: When there's a new message on Slack, create file sharing on Dropbox
When there's a new message on slack, create files on dropbox
Submit
Predicted flow based on your submitted input:
slack
created rawmessage
dropbox
create files

**Figure 1: User interface for our natural language to flow system. The system provides an example of an utterance, a text input area and a submit button. Upon submitting an utterance, a flow is generated and displayed below the utterance. Flows consist of connectors that have applications, actions, and objects.**

the parsing and matching stages – the job of the third and final ordering stage is to enforce what is detected in the first two.

*3.3.1 Parsing.* The first stage of the system parses the utterance to determine candidate components using the Abstract Meaning Representation (AMR) [8, 9, 89]. The result of the parsing is a directed graph of the utterance structure with words labeled with their part of speech. A useful property of an AMR-based parse is that it loses syntactic variances in utterances for the same content, meaning that we are likely to get the same parse for different ways of describing the same flow in natural language [9]. From the parse tree, the system extracts a triple: a candidate component, described by an application name (e.g. "Gmail"), along with their mode of operation (e.g. "create new"), the business object on which it will operate (e.g. "an email object"), and whether the triple is likely to be the trigger or an action. For example, for the utterance "When there is a new message created on Slack, create files on Dropbox", the parsing step would produce two candidate components: a trigger with the application "Slack," object "message" and operation "created," and an action with the application "Dropbox," object "files" and operation "create." During the parsing step, among the parts of speech detected in the AMR parse, the system also looks for common patterns that indicate preferred orders using words like "from", "to", "and", "then", etc (e.g. *"Copy all my files from Dropbox to Google Drive"*, or *"Send an email and then message me if I get a response"*). These are commonly used patterns in the IFTTT[6] (if-this-then-that) paradigm. The candidate component terms and possible orderings are passed to the matching step. In the parsing step, we expected users to need to understand that the system is parsing natural language to determine the structure of the flow, including parts of speech and which components are triggers and actions. Further, our system requires an application name to create a candidate component and, due to limitations of the AMR model, works best if the application name, object, and operation are mentioned explicitly in the utterance.

[6]IFTTT: https://ifttt.com.

*3.3.2 Matching.* The matching step considers each candidate component and attempts to match it to component meta-data stored in a knowledge graph – this stage is crucial to matching the (successfully parsed) items the user mentioned to concepts (i.e. available services and their properties) that the system understands. The knowledge graph contains descriptions of applications, including the objects and operations available for each application. The knowledge graph also contains latent links, which are connections between similar pieces of information created using embedding distance and graph neural networks [31, 112]. For each candidate component, the system attempts to match it to a component available in the knowledge graph and passes this set of components to the ordering step. In the current system, the matches returned from the knowledge graph may include technical terms. Future knowledge graph implementations could also provide terms in more general forms for non-technical audiences. For example, from the candidate components discussed above, the top matches are two components: Slack create message (trigger) and Dropbox create file (action). The system can also provide the five closest component predictions for each candidate, along with a relevance score and meta-data. For the matching step, we expected that users would need to understand that there is a matching happening between their utterance terms and a set of information in the system. However, there is not a specific set of terms that must be used.

*3.3.3 Ordering.* The final ordering step, implemented using an open-sourced AI planning service [85], receives the detected triples along with detected orderings and enforces all this information in the form of a constraints satisfaction problem. The planner uses the candidate components from the parsing and matching steps and API specifications of the inputs and outputs of component actions to sequence the candidate components into a flow based on four considerations: the detected trigger, the detected components, any detected orderings, and the outputs of one component that can be piped to the input of another (e.g. if components A and B are equally likely but B produces something that is an input to A, then the flow A $\rightarrow$ B is less likely than B $\rightarrow$ A). If a trigger is detected, then it is interpreted as a total order between the trigger component and the rest of the flow.[7] The trigger then is always followed by the other actions, though the trigger does not need to be described first. For example 'When there is a new message created on Slack, create files on Dropbox' and 'Create files on Dropbox when there is a new message created on Slack' will produce the same flow. We expected that users would benefit from understanding that the ordering and structure of the generated flow is based primarily on the way the sentence is written and meta-data about the connectors, rather than the order of words in the sentence.

## 4 EXPLANATION TYPES

We designed our explanations based on the information demands of users of intelligent systems [73, 76, 115]. Researchers have found that there are common sets of information needs, such as 'why' and 'how' systems work, the types of inputs and outputs systems handle, the performance of the system, the impacts of changing an input ('what if'), and the limitations of systems. Our explanations aim to address these needs. In this work, we compared our *baseline* system with six explanation types: three system-based (*system-map*, *system-top* and *system-both*), and three social explanations (*social-add*, *social-remove*, and *social-both*).

### 4.1 Baseline

In the baseline condition (see Figure 1), participants only saw the generated output flow. The baseline condition shows no additional explanation of the system behavior or usage.

### 4.2 System Explanations

Our system explanations provide information to the user based on how the system processes the user's utterance to generate a flow. They were inspired by prior work on the design of explanations and common techniques for explanations in generative and natural language systems [30, 115, 132]. The system explanations focus on the 'why' and 'how' of our system in two ways: 1) highlighting relevant natural language in the *system-map* explanation, and 2) providing alternative outputs with confidences in the *system-top* explanation.

#### 4.2.1 *System-map.*

Users of intelligent systems want to understand how the inputs are processed [76, 115]. Our *system-map* explanation is based on explanations that highlight inputs in generative code translation systems [115, 132]. Highlighting, or visually emphasizing the elements of input relevant to the produced output, is a common approach in natural language and generative systems [30], often using attention distribution [124] or an attention mechanism [88]. Attention distribution was selected as one of three main methods of explaining generative AI systems and was described favorably by users [115].

In the *system-map* explanation (see Figure 2-1), the user's input utterance is reproduced above the generated flow with colored highlighting. The color of the text highlighting matches the border around the associated component. The words to highlight are selected based on the parse of the user's utterance. They are the words the system uses to match to the knowledge base to find the application names, objects, and operations for the flow components.

#### 4.2.2 *System-top.*

Users of intelligent systems also want to understand the potential outputs of a system and the system's certainty or performance [76]. Our *system-top* explanation is similar to prior systems showing alternatives and model confidence in explanations. In a generative AI system, users wanted to know about the other outputs the system could have generated [115]. Providing the confidence of AI models is also a common and beneficial explanation method for users in generative AI systems [115, 132], as well as decision-based machine learning systems [18].

In the *system-top* explanation (see Figure 2-2), the system displays the top five predicted components for each component in the generated flow. These can be considered alternatives, because they are the next most likely components the system would have selected. These top predictions are displayed in boxes below the generated flow and are outlined in colors matching the corresponding components in the flow. Alongside the name of each predicted component is the relevance score, or the closeness of the match.

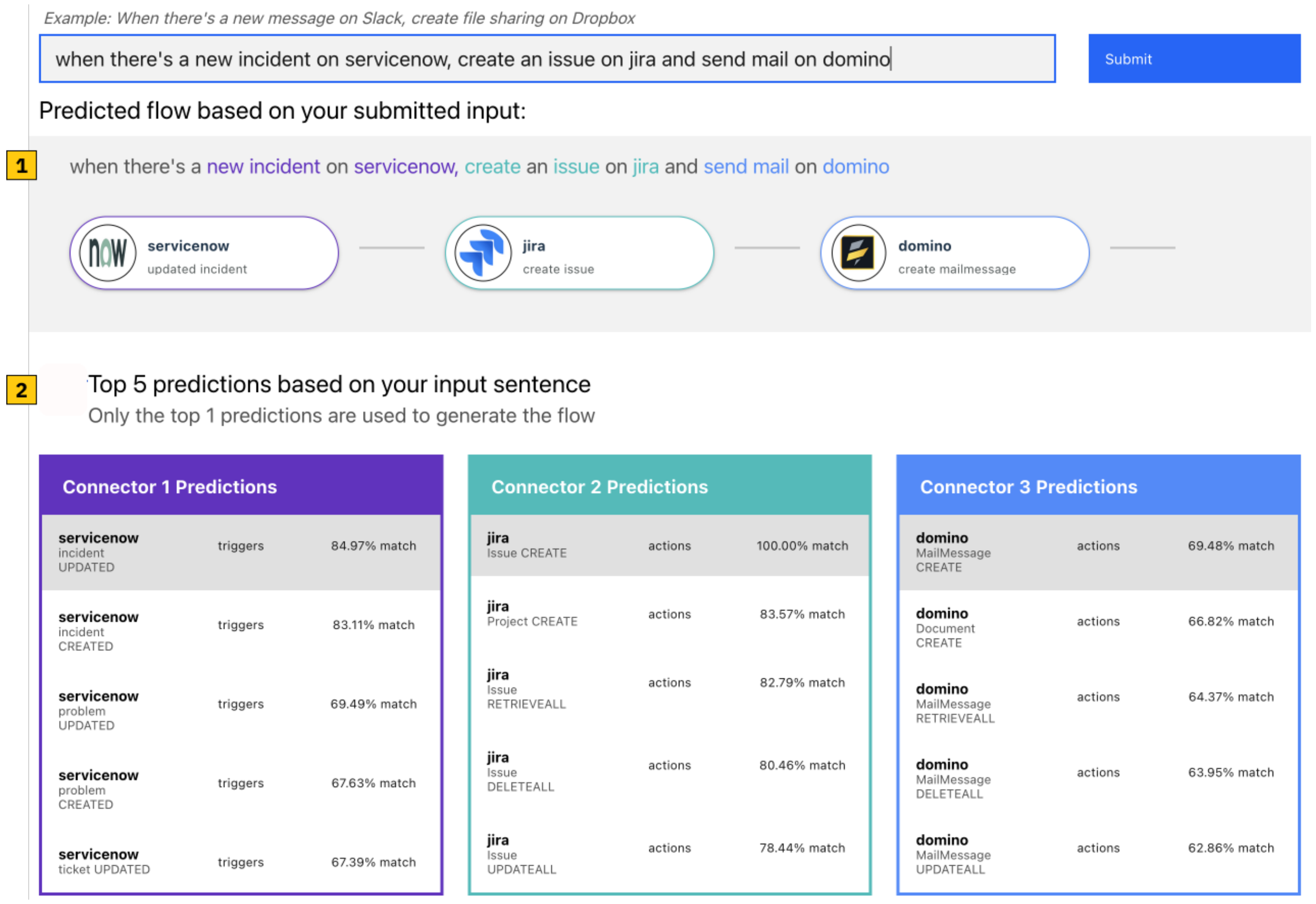
#### 4.2.3 *System-both.*

The *system-both* explanation provides both the *system-map* explanation and the *system-top* explanation, as shown in Figure 2. We chose to compare an explanation that included both types of system-based explanations, as the two types of explanations provide potentially complementary information. The *system-map* explanation might indicate to a user the words that they need to change or remove from their utterance, while the *system-top* explanation might provide insight into words users could add. Providing both types of explanations also has a risk that users may be overwhelmed by the amount of information provided, or perceive the cost of reading the explanation as too high [22].

### 4.3 Social Explanations

Our social explanations provide users with information about how other users modified their utterances to move closer toward their goal flow. These explanations address the 'what if,' 'how to,' and 'limitations' information needs [76]. We designed our social explanations leveraging recent work exploring the design of explanations based on usage and context [36, 115]. Research on an AI decision support system explored explanations that showed whether other users took system recommendations [36]. In the context of a generative AI system for code, an open design probe showed that users wanted to know about other users' experiences with the AI system, including what the AI can do and what it can't, especially in cases where others had similar tasks or needs [115]. Thus, we designed our social explanations to make suggestions to users on how they can change their natural language to improve their outcome. The core idea is also related to counterfactual explanations [123, 141] that show how different inputs might change a decision boundary,

[7] Note that this can lead to an over-constrained system where the detected trigger conflicts with a detected ordering from the user utterance. In such cases, the processing pipeline runs itself again while ignoring the parsed orderings, thereby prioritizing the trigger constraint.


Example: When there's a new message on Slack, create file sharing on Dropbox

when there's a new incident on servicenow, create an issue on jira and send mail on domino

Predicted flow based on your submitted input:

1 when there's a new incident on servicenow, create an issue on jira and send mail on domino

2 Top 5 predictions based on your input sentence

Only the top 1 predictions are used to generate the flow

| Connector 1 Predictions | | |
|---|---|---|
| **servicenow** incident UPDATED | triggers | 84.97% match |
| **servicenow** incident CREATED | triggers | 83.11% match |
| **servicenow** problem UPDATED | triggers | 69.49% match |
| **servicenow** problem CREATED | triggers | 67.63% match |
| **servicenow** ticket UPDATED | triggers | 67.39% match |

| Connector 2 Predictions | | |
|---|---|---|
| **jira** Issue CREATE | actions | 100.00% match |
| **jira** Project CREATE | actions | 83.57% match |
| **jira** Issue RETRIEVEALL | actions | 82.79% match |
| **jira** Issue DELETEALL | actions | 80.46% match |
| **jira** Issue UPDATEALL | actions | 78.44% match |

| Connector 3 Predictions | | |
|---|---|---|
| **domino** MailMessage CREATE | actions | 69.48% match |
| **domino** Document CREATE | actions | 66.82% match |
| **domino** MailMessage RETRIEVEALL | actions | 64.37% match |
| **domino** MailMessage DELETEALL | actions | 63.95% match |
| **domino** MailMessage UPDATEALL | actions | 62.86% match |

**Figure 2: Input, predicted flow and system explanations. The interface shows *system-both*, which includes what users see for *system-map* (1) and what users see for *system-top* (2)**

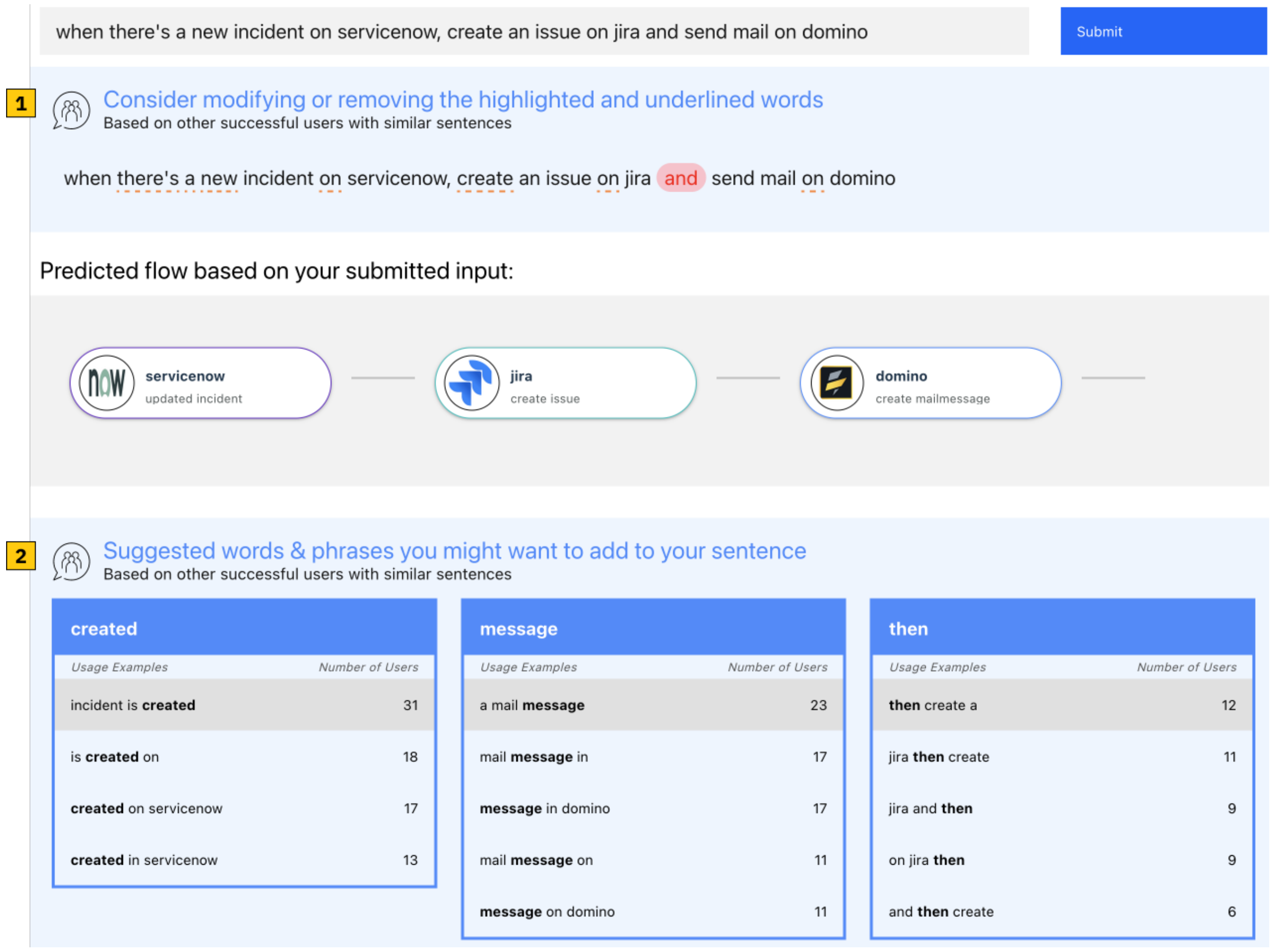
though counterfactual explanations are not necessarily social in nature. We first describe the *social-add*, *social-remove*, and *social-both* explanations, followed by the the data we used in the explanations, and how we decided if a user was moving closer to their goal flow.

#### 4.3.1 *Social-remove: suggesting words to modify or remove.*

The *social-remove* explanation (see Figure 3-1), suggests that users may want to modify or remove words by highlighting and/or underlining those words (in this case, the system highlighted 'and', and underlined 'there's', 'a', 'new', 'on', 'create'). We selected the words to highlight and underline using two sets of words: 1) top five words users frequently removed, and 2) top five words more common in unsuccessful utterances than successful ones. If a word was in both sets of words, the explanation highlighted it, while if it was only in one list, the explanation underlined it. We selected the top five words frequently removed by finding the words users removed that improved the success of the following utterance (based on the scoring algorithm described below). We weighted the frequencies by the score of the resulting flow, as a more successful flow may mean that removing that word was more beneficial. We selected the top five words more common in unsuccessful utterances than successful ones by comparing the frequencies of words in unsuccessful and successful utterances. We chose the number of words to highlight/underline and designed the *social-remove* highlighting and underlining to closely resemble the *system-map* explanation, in order to reduce the impacts of the amount of information and design. For our system, common English words like 'and', 'on', and 'new' impact the content, structure, and ordering of the generated flows. Other systems that do not leverage these common words may want to remove these from suggestions.

#### 4.3.2 *Social-add: suggesting words and phrases to add.*

In the *social-add* explanation (see Figure 3-2), the user is provided with a set of words and three-word phrases they might want to add to their utterance, based on other users' data. Specifically, the interface suggests up to three words a user might want to add to their utterance and for each word, up to five three-word phrases other users have used in successful utterances. To suggest words that a user might want to add to their utterance, we used 1) common three-word phrases in successful utterances, and 2) words users commonly added that led to more successful flows. In order to find the three-word phrases



when there's a new incident on servicenow, create an issue on jira and send mail on domino

Submit

1

Consider modifying or removing the highlighted and underlined words

Based on other successful users with similar sentences

when there's a new incident on servicenow, create an issue on jira and send mail on domino

Predicted flow based on your submitted input:

servicenow — updated incident

jira — create issue

domino — create mailmessage

2

Suggested words & phrases you might want to add to your sentence

Based on other successful users with similar sentences

**created**

| Usage Examples | Number of Users |
|---|---|
| incident is **created** | 31 |
| is **created** on | 18 |
| **created** on servicenow | 17 |
| **created** in servicenow | 13 |

**message**

| Usage Examples | Number of Users |
|---|---|
| a mail **message** | 23 |
| mail **message** in | 17 |
| **message** in domino | 17 |
| mail **message** on | 11 |
| **message** on domino | 11 |

**then**

| Usage Examples | Number of Users |
|---|---|
| **then** create a | 12 |
| jira **then** create | 11 |
| jira and **then** | 9 |
| on jira **then** | 9 |
| and **then** create | 6 |

**Figure 3: Input, predicted flow and social explanations. The interface shows *social-both*, which includes what users see for *social-remove* (1) and what users see for *social-add* (2)**

to suggest, we split successful utterances into trigrams (sets of sequential three words) and calculated the frequency of the trigrams. We then found the set of frequent trigrams that were not in the user's utterance, but had an overlapping word with the utterance. To narrow down which trigrams to recommend, we selected three recommendation words and grouped the trigrams by those words. We selected these three words by finding the top words users added to their utterances that improved flow scores, and weighted them by the average score of the utterance after adding them. For each utterance, the explanation then recommended the most frequent words not in the user's utterance that had common successful trigrams. The *social-add* suggestions were presented in a similar style and format to the *system-top* explanation in order to control for the amount of content presented and the style of presentation.

#### 4.3.3 Social-both.

The *social-both* explanation includes both the *social-add* and the *social-remove* explanations (as shown in Figure 3). Like the *system-both* condition, we explore providing both types of social explanations together, as they provide complementary explanation information, but also risk overwhelming users with too much information.

#### 4.3.4 Data and Scoring.

In order to create explanations based on other users' data, we needed a set of user utterances. We acquired these utterances by first collecting data for the *baseline*, *system-map* and *system-top* explanation groups. To generate an explanation for a user's utterance, we took the utterances from our set of data that were for the user's particular task, in order to focus the explanation on the user's context. Our explanations use correct utterances, incorrect utterances, and pairs of utterances in which the changes move users closer to their goal flow. In order to establish which utterance changes moved users closer to their goal, we scored output flows with up to 12 points by comparing the generated flow to the task flow. For each component in the generated flow, we added to the score: 1 point per matching application, 1 point per matching object, 1 point per matching operation, and 1 point per matching location. During this scoring, we maximized the score by matching connectors in the generated flow to those in the task flow such that those with the most matching features were matched first. We then reduced the score by 2 for each extra component beyond the number in the correct flow.

## 5 STUDY METHODS

We ran a between-subjects study to compare explanation methods for our natural language to flow system.

### 5.1 Procedure

Participants 1) completed informed consent, 2) viewed instructions, 3) completed a training task, 4) completed three tasks with or without an explanation, and 5) completed a survey. Participants were randomly assigned to one explanation condition (*baseline*, *system-map*, *system-top*, *system-both*, *social-add*, *social-remove*, or *social-both*).

The tasks asked users to write a natural language sentence to try to generate a provided flow. The task flow was provided as an image (as shown in Figure 4). All users had an input textbox and an example utterance as shown in Figure 1. The example utterance was purposefully very similar to the training task, but unlike any of the three main tasks. Users had up to five attempts to try to generate the correct task flow by inputting or modifying an utterance. Upon submitting an utterance, the system would generate a flow, displayed visually to the user in the same way as the tasks. This enabled users to compare the generated flows to the task. If an utterance generated a correct flow sooner than the fifth attempt, the system moved the user on to the next task. If participants were in a condition with an explanation, an explanation was shown with each generated flow.

*5.1.1 Tasks.* All participants had the same training task first and then three main tasks. Participants were distributed across the nine possible orderings of the three tasks, to reduce the impact of learning on the success of an individual task. All participants had no explanation for the training task, which was designed to familiarize participants with the system prior to introducing explanations. We did not analyze the data from the training task.

The three main tasks (see Figure 4) were designed to have similar and high difficulty: they each included three flow components. Participants saw explanations for these tasks based on their condition. The three main tasks asked participants to try to generate business automation flows, in which an event triggers two actions in sequence. The events and actions all include business applications and data operations, like creating an issue in Jira. Our system was designed to enable non-technical users to create these automation flows using natural language. Since we ran our study using Mechanical Turk, we aimed to select components for the tasks that were accessible to non-business users as well. We designed the three tasks to include different application names and details to reduce overlap in the vocabulary used for each utterance.

*5.1.2 Survey.* In the survey we collected: demographic information, prior experience with natural language systems, programming experience, agreement on a 5-point Likert scale with six statements about the system to capture users' mental models (see Table 2), and responses to three open-ended questions (two about how the system works and one about use of the interface). We designed our mental model questions based on prior work on mental models of AI systems and the conceptual model of our system [44, 65].

### 5.2 Participants

We recruited participants on Amazon Mechanical Turk who met the following requirements: had a HIT approval rate greater than 95% and were located in a country where English is the primary language. We recruited four participants per condition who were Master Turkers, but stopped requiring this due to slow recruitment. We paid participants 4 USD (more than federal minimum wage for a 15-20 minute task) and provided a 10 cent bonus for each attempt users had left when they got the task correct, as an incentive, with a maximum bonus of $1.60. Due to known issues with Mechanical Turk work quality, we filtered our data, removing participants who did not pass our survey attention check, had a non-English primary language, did not follow directions, or had more than three system errors. We analyzed data from a total of 252 participants, with 42 in *baseline*, 40 in *system-map*, 40 in *system-top*, 37 in *system-both*, 33 in *social-remove*, 29 in *social-add*, and 31 in *social-both*. Participants rated their programming experience on average as 3.9 out of 10 (SD = 2.9, Median = 3). About half of participants had rarely interacted with a natural language (NL) system, either 1-4 times (44%) or never (9%). The other half of participants had interacted with an NL system more often, either 4-10 times (20%) or 10 or more times (27%). Most participants (65%) had undergraduate degrees, while some had high school or less (25%) and some had graduate degrees (10%).

### 5.3 Measures

We collected and analyzed the following measures:

- **Correctness**: A task was correct if it matched the task flow or incorrect if it did not.
- **Correctness in repair**: Repair measures whether a task that took more than one attempt was correct.
- **Efficiency**: The number of attempts it took users to get tasks correct.
- **Use of explanations**: Changes in utterances between attempts, whether the words added and/or removed were part of an explanation, and survey responses about parts of the interface that helped users.
- **Mental model**: Six mental model questions and users' descriptions of how they thought the system worked.

### 5.4 Analysis

We performed quantitative analysis on correctness and efficiency measures and mental model survey questions and qualitative analysis on participants' open-ended responses.

We used generalized linear mixed models (GLMM) and Wald chi-squared tests to evaluate the effect of explanation type on correctness, efficiency, and mental model questions [40, 63]. Condition, programming experience, natural language system experience and education level were fixed effects and user was a random effect. We also used task as a fixed effect for correctness and efficiency and question as a fixed effect for the mental model questions. We used

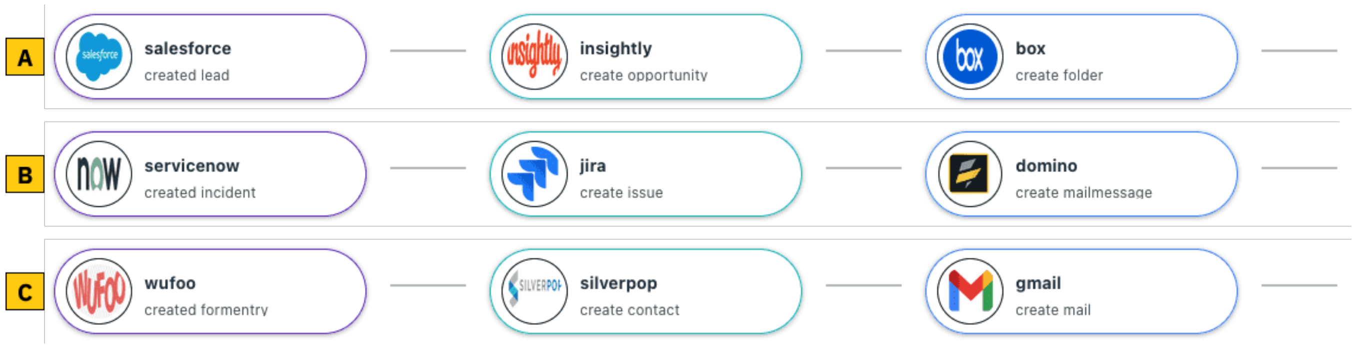


**Figure 4: The three main tasks. Participants saw the image flow for each task.**

the emmeans [8] package for pairwise post-hoc tests using the Tukey method to adjust for multiple comparisons [55].

For the qualitative analysis, we used an iterative thematic analysis approach [20] with inter-rater agreement. Two authors independently read through participants' responses to the three open-ended questions, using an open coding approach. The authors then discussed the codes and themes, coming to agreement on the set of codes. The authors independently applied the codes to 20% of the data. Using Cohen's Kappa with Jaccard distance due to multiple codes per item, we had an inter-rater agreement of $\kappa = 0.77$, which is considered substantial agreement. The authors then independently coded the remaining data.

## 6 RESULTS

We answer our research questions: 1) how do explanations impact repair success, overall correctness, and efficiency, 2) how did participants use explanations, and 3) how did explanations impact users' mental models of the system?

### 6.1 RQ1: How do explanations impact repair success, overall correctness, and efficiency?

*6.1.1 Ability to repair: correctness after first attempt.* We expected explanations to help participants the most in repair, as they received an explanation for each task only after generating at least one incorrect flow. We found a main significant effect of condition ($\chi^2(6) = 21.7, p < 0.01$) and task ($\chi^2(2) = 20.2, p < 0.001$) on repair correctness (see Figure 5).

Post-hoc tests showed that participants with *social-add* were able to repair more successfully than those in the in the *system-map* ($p < 0.01$), *system-top* ($p < 0.05$), and *social-remove* ($p < 0.05$) conditions. Participants with *social-both* explanations were also significantly more successful at repair than those with *social-remove* ($p < 0.05$) and marginally more successful than those with *system-map* ($p = 0.05$). Though we expected the three tasks to have similar difficulty, participants had higher repair success rates in task A than task B ($p < 0.01$) and task C ($p < 0.001$). Note, participants saw tasks A, B, and C in different orders to balance for the effects of learning.

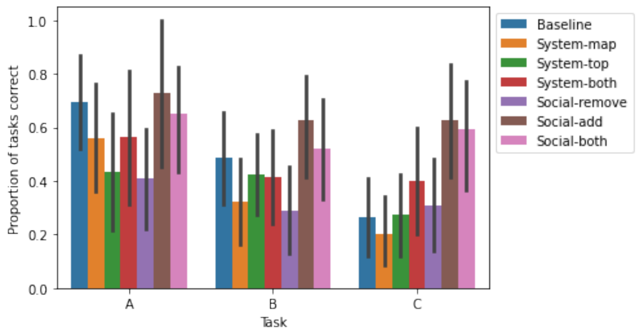


**Figure 5: Percentage of participants who got each task correct per condition after the first attempt**

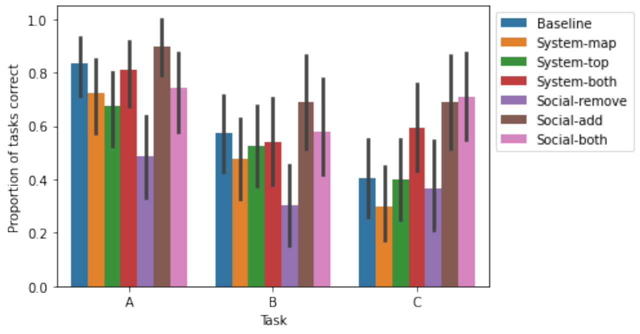


**Figure 6: Overall percentage of participants who got each task correct.**

*6.1.2 Overall correctness.* Task correctness tells us whether participants were able to use natural language to generate their goal flow, including on a first attempt. This measure evaluates if explanations impacted users' abilities to use the system effectively even before needing to repair. An improved mental model based on an explanation might lead users to only need one attempt on subsequent tasks. We found significant main effects of condition ($\chi^2(6) = 25, p < .001$) and task ($\chi^2(2) = 48.6, p < .001$) on overall correctness. There were also marginal effects of prior experience

[8]https://www.rdocumentation.org/packages/emmeans/versions/1.8.1-1

**Table 1: Mean±standard deviation words added and removed per condition from explanation, not from explanation (other), and overall. Commonly added and removed words are bolded if they appeared in explanations for that condition.**

| Condition | Removed from explanation | Removed Total | Common words removed | Added from explanation | Added Total | Common words added |
|---|---|---|---|---|---|---|
| Baseline | n/a | 3.0±3.4 | create(65), a(52), on(36), and(35), in(33), to(30) | n/a | 3.2±3.6 | create(85), a(64), on(49), in(45), new(34), when(33) |
| System-map | 1.1±1.4 | 3.4±3.9 | **create(100), a(74)**, on(61), **to(44), new(39)**, then(38) | 0.5±0.9 | 3.5±3.7 | **create(103)**, a(77), on(70), and(42), to(41), **new(39** |
| System-top5 | 0.3±0.6 | 3.3±4.0 | **create(85)**, a(56), to(49), in(47), then(45), and(42) | 0.3±0.7 | 3.5±3.7 | **create(96)**, a(60), in(54), **created(54)**, to(53), and(48) |
| System-both | 0.2±0.5 | 2.5±2.8 | **create(65)**, on(36), a(29), mailmessage(23), new(20), and(19) | 0.2±0.5 | 2.7±3.0 | **create (58)**, a(37), on(28), new(26), in(25), **message(25)** |
| Social-remove | 1.4±2.0 | 3.3±3.5 | **create(81), a(64), on(59), and(46), created(39), in(38)** | 0.4±0.8 | 3.3±3.3 | **create(78), on(56), a(52), and(39), created(38), message(34)** |
| Social-add | 0.6±1.3 | 2.6±3.0 | **a(43), in(36), create(31), new(28), on(25), and(22)** | 1.0±1.6 | 2.7±3.0 | **on(50), a(41), create(36), created(32), then(29), in(27)** |
| Social-both | 1.1±2.0 | 2.9±3.2 | **create(73), a(42), on(41), in(38), and(33), then(28)** | 1.0±1.7 | 3.0±3.3 | **create(74), on(62), a(49) in(34), created(34), and(32)** |

with natural language systems ($\chi^2(3) = 7.5, p = .06$), and education level ($\chi^2(3) = 6.9, p = .08$). Post-hoc tests showed that, like for repair, *social-add* had significantly higher task correctness than *system-map* ($p < .05$), *system-top* ($p < .05$), and *social-remove* ($p < .001$). *Social-both* also had significantly higher task correctness than *social-remove* ($p < .05$). Again, for task, participants got significantly more task A correct than B ($p < .001$) and C ($p < .001$). Figure 6 shows the percentages of users who got each task correct per condition and task. Overall correctness closely followed the results of repair, though some personal qualities may have also had an influence in overall success.

### 6.1.3 *Efficiency: attempts to success.*

The number of attempts gives us an understanding of how efficient users were in generating correct flows. Participants could take up to five attempts for each task. We did not want the number of incorrect tasks, which always had five attempts, to skew this analysis, so we only compare the number of attempts for tasks that were correct. We found a significant main effect of task on number of attempts to success ($\chi^2(2) = 15.2, p < 0.001$). The post-hoc follow-up test showed a significant difference between tasks A and B ($p < 0.001$) and a marginally significant difference in the number of attempts between tasks A and C ($p = 0.08$).

## 6.2 RQ2: How did participants use explanations?

To answer this question, we analyzed how participants modified their utterances and how participants talked about using the elements of the interface to author their utterances.

### 6.2.1 *How did participants modify their utterances?*

To understand how participants used explanations, we explored how often participants added and/or removed words mentioned in the explanations. We used Python's *difflib* library [9] to analyze the words added and removed between successive utterances. Due to the differing nature of the explanations available, we descriptively report the average number of words added and removed per condition, how many of those words appeared in the explanation provided, as well as common words added and removed (see Table 1). The bold words appeared in explanations for that condition.

For the *system-map* and the *system-both* conditions, we would expect that participants might remove or move highlighted words to try to correct their utterances. We did find that participants in *system-map* removed on average 1.1 words that were highlighted (SD=1.4). However, those in *system-both* only removed on average 0.2 highlighted words (SD=0.5). For the *system-top* condition, we might expect participants to add words from the presented top five connectors, though participants only added on average 0.3 words displayed in the explanation (SD=0.7). The system explanations focused on the actions, like 'create' and 'created', as well as the entities, like 'message' or 'mailmessage'. However, many of the words changed in the utterances were words that impacted the structure of the utterance, like 'a', 'on', 'and', and 'then'. Some words were frequently both added and removed, like 'create.' This may have occurred because users were experimenting with adding and removing the word throughout their attempts or because they needed to change where in an utterance a word was located. For example, users often added 'create' to their utterances, as this was a common operation in the tasks. However, they also often needed

[9]https://docs.python.org/3/library/difflib.html

**Table 2: Mental Model Questions, rated on a Likert Agreement scale 1: Strongly Disagree, 5: Strongly agree**

| Question (shortened) | Agreement |
|---|---|
| T1-part-speech: The system looks for and uses certain parts of speech | M=4.1, SD=0.7 |
| T2-match: The system attempts to match terms to a set of application, action, and object names. | M=4.1, SD=0.7 |
| T3-app: The system requires application names | M=4.1, SD=0.9 |
| F1-component: One action or task always corresponds to one flow component | M=3.4, SD=0.9 |
| F2-specific: There is one set of specific keywords, all of which must be in the sentence. | M=3.5, SD=1.0 |
| F3-order: The order of the input is always going to relate to the order of the flow. | M=3.7, SD=1.0 |

to replace 'create' with 'created', because the flows are triggered by a first operation like a message that was created. Participants in the social conditions added and removed around one word from the explanations across the three conditions. However, for all conditions, participants added and removed close to two words not from explanations, based on the total numbers of words added and removed.

*6.2.2 What information helped participants to use the system?* The survey asked participants if they used any particular information in the interface or prior knowledge to help them in using the system. Our qualitative analysis showed that participants most often talked about the example sentence (33%) and occasionally referenced the instructions (4%) which were both available in all conditions. Participants talked about using the two main forms of feedback from the system: explanations (13%) and the flow diagrams generated by the system (17%). Participants also talked about prior experiences helping them, including trial and error with the system (11%), as well as prior knowledge from other experiences like programming classes (15%).

## 6.3 RQ3: How did explanations impact users' mental models of the system?

*6.3.1 Mental model survey questions.* In the survey, participants responded with their agreement (strongly disagree to strongly agree) for six statements about the system (see Table 2). Three of the statements were correct (T1-T3) and three were incorrect (F1-F3). We hypothesized that the explanations would impact users' understanding of how the system works. However, we did not find significant differences across explanation types for the mental model questions. There was a significant effect of prior experience with natural language systems ($\chi^2(3) = 13.5, p < .01$) and of question ($\chi^2(5) = 1420.1, p < .001$). There were also marginal effects of programming experience ($\chi^2(9) = 15, p = .09$) and education level ($\chi^2(3) = 6.9, p = .08$). Post-hoc tests showed that those who have interacted with natural language systems 10 times or more had significantly stronger correct agreement with mental model statements than those who had interacted with systems 4-10 times ($p < .01$) and marginally stronger correct agreement than those who interacted with natural language systems 1-4 times ($p = 0.08$). Participants were more likely to agree with the true statements and less likely to agree with the incorrect statements. We found participants' agreement with each true statement was significantly higher than their agreement with each false statements ($p < .001$). F3-ordering also had significantly higher incorrect agreement than F1-component ($p < 0.01$) and marginally higher than F2-specific ($p = 0.08$). Figure 7 shows the overall number of participants who agreed or disagreed with each statement.

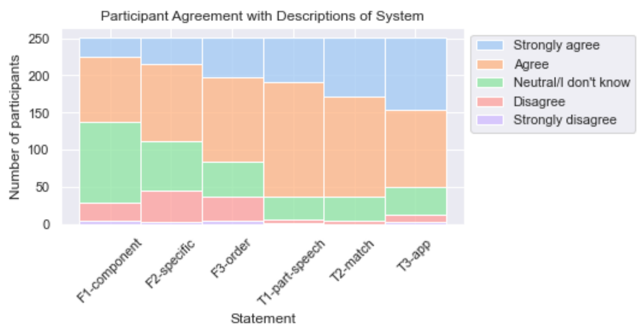


**Figure 7: Number of participants who chose each agreement level for the six statements describing the system**

*6.3.2 How did participants describe how the system works?* We asked participants to describe how they thought the system translated their natural language to a flow and if they used any particular words or language that they thought helped them succeed. Table 3 shows our thematic analysis codes for participants' responses to these questions and the percentages of participants who mentioned each code. We grouped the types of responses by correctness, similar to prior work on mental models of AI systems [65].

We found that participants most often talked about keyword matching (59% of participants), followed by structuring (38% of participants). Participants often seemed to correctly understand that the system was attempting to match particular words in their utterances to a set of words within the system. Some also seemed to correctly understand that their utterance needed to have a particular structure (such as When X then Y) in order for the system to parse it correctly, though this was less common. Much more rarely, a few participants also talked about the system needing related words to be close together and that the system was splitting their sentences into parts and processing each of those parts. Some participants talked about the system needing simple input (26% of participants). In some ways, this is correct, as including extraneous or unnecessary details can result in more errors. However, there is a limit to this, as the system also needs some form of structure from the natural language itself - if the input is too simple or phrased like code, the system will have a hard time processing it.

**Table 3: System Understanding: Thematic Analysis Codes and Themes**

| Correct? | Code | Description | % users |
|---|---|---|---|
| Correct | keyword matching | The system has a knowledge base, recognizes certain keywords (app names, actions verbs, etc.), ignores stop words or words that it can't recognize | 59% |
| Correct | structuring | The system recognizes certain words that are used to create the structure and ordering of the flow such as if/then, when/then, comma, and | 38% |
| Correct | connecting words | Tie words together, place them near each other | 4% |
| Correct | divide and conquer | The system splits the sentence into small parts | 4% |
| Partial | simple input | Can only recognize machine-like simple literal direct instructions | 26% |
| Partial | writing style | Use of passive voice, past/future tenses, specific, descriptive, normal language styles, etc. | 8% |
| Partial | problematic words | Some words (e.g., connecting words, nouns) are prone to errors | 4% |
| Incorrect | random component | A new component was created for no reason | 9% |
| Incorrect | left-to-right ordering | Order the flow in the order of which the words appear in the sentence (left to right) | 9% |
| Incorrect | ignored ordering | The system ignored the ordering | 1% |
| N/A | uncertain | Don't know, uncertain, confusing | 14% |

Participants most often had misconceptions about the way the system handled ordering (10% of participants) and about the generation of components (9% of participants). Participants either thought that the system took the ordering of a sentence from left-to-right to generate a flow, or ignored the ordering entirely. Participants also talked about random components being generated that they did not ask for. This was caused by issues in the matching between terms in the utterance and terms in the knowledge base, which participants did not seem to understand.

## 7 DISCUSSION AND DESIGN RECOMMENDATIONS

We discuss design recommendations for explanations, future directions for understanding and improving users' mental models, and task design considerations for evaluating explanations for natural language systems.

### 7.1 Exemplary and Contextual Explanations

Our results showed differences in the effectiveness of explanations for supporting repair and overall correctness in usage of a natural language system that generates flows. In particular, *social-add* and *social-both* better supported users than *system-map*, *system-top*, and *social-remove*. While our *social* explanations were inspired by existing work on social explanations that show other users' interactions, they were also more concrete than the system-focused explanations. Thus, the improvements we saw from our *social* explanations may be more due to the concrete nature of these explanations. In particular, our results indicate that systems can support users by: 1) providing examples of successful utterances, and 2) providing task or context specific support.

One way to support users in a natural language system may be to provide examples of effective natural language. Our *social-add* explanations showed words and phrases that other successful users included in their utterances. Further, 33% of participants reported that they used the provided static example sentence to help them author their utterances. In contrast, *system-map* and *social-remove* explanations draw attention to words that a user may want to modify, without suggesting how. *System-top* provided top predictions for flow components, which provides information about the vocabulary the system uses for components, but does not provide natural language structure or style. Participants may have tried to remove problematic terms from their utterances based on these explanations without replacing them with helpful terms. Further, in the *social-remove* condition, there was no guarantee that the explanation correctly pointed out what needed to change.

Showing users successful natural language utterances aligns with the many systems that provide auto-complete [17, 32, 54, 110, 121, 138] and example or suggested utterances [29, 32, 42, 114]. *Social-add* explanations also answer 'how to' questions, which have been identified as a user need [73, 115]. In contrast with common support, we showed that providing information about how other users *modified* their utterances was helpful to users in repair, somewhat similar to counterfactual explanations [123, 141] or example-based explanations [116] in decision systems. Future work could explore the types of exemplary support and the contexts in which they work best.

While participants seemed to understand at a high level that our system attempts to match terms from their utterances to a knowledge base, it was often not enough to help users succeed. In particular, a common issue users had in tasks B and C was using terms from the provided task flows, like 'formentry' or 'mailmessage' in their natural language utterances. Instead, users needed to break these terms into natural language, like 'form entry' or use synonyms that exist in dictionaries like 'email'. Users' general mental model of keyword matching often did not help them resolve these issues. Our *social* explanations were task-specific and communicated the system's 'limitations' [115]. We implemented task specificity in our *social* explanations by generating an explanation for a particular utterance using data from previous users for the same task. This method controlled whether the information shown to users was actually relevant for their task in our study. However, a real system doesn't know the user's task, so it needs a way to make an educated guess about the data it should use in a social explanation. One way to do this could be to cluster previous users' utterances and find the cluster of prior user utterances closest to the current user's input. Then, the system can use the cluster of utterances to make suggestions to the current user.

## 7.2 Mental models: identify gaps and tailor

We found effects of users' prior experiences on mental models, which aligns with prior work that personal characteristics can impact mental models of recommendation systems [86, 93]. However, we were surprised to find no effect of explanations on users' mental models, despite differences in prior work [7, 105]. The impact of users' prior experiences with natural language systems may have overpowered the impact of explanations in our short study.

Participants often understood that the system was matching words in their utterances to a knowledge base. This matching is common in natural language systems that the general population interacts with on a regular basis, like search [10], chatbots [2, 142], and personal voice assistants [143]. Our *system* explanations may have primarily reinforced ideas about keyword matching. For example, the *system-map* explanation mapped the words from a user's utterance that were matched to the knowledge base to select each component in the flow. The *system-top* explanation provided the top matches from the knowledge base. In contrast, the *social* explanations suggested words to add or remove from utterances. This included not only the core terms used by the system to match to components, but also words that impacted the structure and ordering of a flow, like 'when', 'then', or 'and.' The *social-add* explanation also suggested phrases, which are more likely to indicate writing style and structure. Participants were much less likely to talk about the trigger-action structure needed, correctly identify how the system ordered output flows, or correctly discuss problematic words or writing styles. These properties are specific to systems like ours, that generate complex multi-step processes, compared to systems commonly available to the general public. One reason participants may have been more successful in the *social-add* and *social-both* conditions is due to the information provided beyond matching.

In order to effectively use a system, users must establish a mental model that matches the specific conceptual model of a system [44]. Another way to think about explanations moving forward may be to explain the differences between a system and other well-known similar systems, as these gaps can lead to misconceptions. For systems with a broad user population, segments of the population may have different gaps in mental models. Researchers have begun to suggest tailoring of explanations, such as based on personal characteristics like need for cognition [86], age-level [93], or style preferences [62]. Our work suggests that future systems might consider tailoring explanations based on users' prior experience with similar types of systems and how a particular system differs from other common conceptual models. By personalizing explanations, systems can reduce the amount of information needed to clarify the critical and unique elements of a particular system.

[10]https://developers.google.com/knowledge-graph/

## 7.3 Task design considerations for evaluating explanations

Our study provides new insight into the evaluation of explanations for natural language and generative systems through a large-scale study in a working system. Much of the prior work in this area has focused on scenario-based explorations and evaluations [45, 115]. However, proxy tasks and subjective measures may not predict actual performance with systems [21]. Our study enables us to compare explanation types in a controlled way and capture users' true performance with the system. We were somewhat surprised that the *system-map* and *system-top* explanations did not help users more. We discuss how our task design may have impacted these outcomes and provide suggestions for future ways to evaluate users' performance based on explanations.

Our tasks asked users to write natural language utterances to describe three provided flows. In designing tasks to evaluate explanations, we considered whether the tasks should ask users to generate a particular flow or a flow of their choosing. We chose to provide specific goal flows, as our Mechanical Turk participants likely had less experience with the applications in our flows than typical users. This means that it might be difficult for users to think of an automation flow to describe. Providing goal tasks, however, may have had an impact on which explanations helped users the most. Imagine a user who is thinking of their own flow. This user may not know all of the possible flow components available, making a *system-top* explanation more useful. Future work should additionally evaluate the impact of explanations in open-ended tasks, where users design their own flows.

Further, in this study, we evaluated three tasks and aimed for them to have similar complexity so that users might be able to learn and improve across the tasks. However, for complex systems, small numbers of tasks may not cover all of the important qualities of a system. One way to assess more tasks without requiring longer study times is to select a small number of tasks for each user from a larger task bank [7]. Having a larger task bank would also support having more tasks of varying difficulty. We found that task A had higher success rates and efficiency than tasks B and C. While the sentence structure and terms in task A were relatively straightforward based on the task and example sentence, tasks B and C were more complicated. Participants often had difficulties in tasks B and C using terms from the tasks that were not recognized as natural language by the system, like 'mailmessage', which needed to be described in a common English term like 'email.' Thus, tasks B and C had lower success rates and were more challenging for participants. Different difficulty tasks may benefit from explanations differently. For example, in Task C, *social-add* and *social-both* had close to double the successful repairs as the other conditions, compared to tasks A and B, which had smaller gaps between the success rates. In addition to creating a broad set of tasks to address the entirety of a system, future work should consider evaluating explanations with a variety of task complexity levels in order to understand the types of tasks where explanations will provide the largest impact for users. These evaluations could also help us to understand whether users would benefit from adaptive explanations, based on the complexity or type of task.

## 7.4 Limitations

Our study has several limitations: our user population, task setup, and the length of the study. While we attempted to make our tasks generic enough that they would be accessible to the general population, our tasks were still business-focused. Many of our participants likely had not heard of some of the applications used. This may have limited the vocabulary they used. Another potential impact on users' natural language was the task setup. In order to ensure that the goal of the task was clear to the general population, we provided the exact flows, including natural language terms, in the tasks. Users could leverage this natural language for their own inputs and sometimes did. However, many participants used diverse natural language, attempting to describe the flows in their own words. Further, the provided flows did not instruct users on how to formulate the sentences into natural language, which was an important aspect of the task. Having users work with our system for only four tasks over a short period of time limited how much we could study the impacts of explanations on better understanding of the system. It is possible that longer exposure to the system and explanations could have led to some explanations having a larger impact on performance or mental models.

## 8 CONCLUSION

In this work, we evaluated the impact of six explanation types on the use and mental models of a working natural language system that generates flows. Our data suggest that our explanations that suggested task-specific terms to improve utterances may be more helpful to users than explanations that focus primarily on suggesting words to modify, demonstrating how terms are used to match to a knowledge base, or demonstrating the confidence of the system. Participants were more able to modify their utterances with *social-add* explanations, got more tasks correct, and used the social explanations more when they modified their utterances. Users will likely benefit from a system that suggests concrete terms and phrases, as well as explanations that are adapted to users' current context and mental model. Our results and recommendations have the potential to support users across the many existing and future natural language and generative systems.

## REFERENCES

[1] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access* 6 (2018), 52138–52160.

[2] Addi Ait-Mlouk and Lili Jiang. 2020. KBot: a Knowledge graph based chatBot for natural language understanding over linked data. *IEEE Access* 8 (2020), 149220–149230.

[3] Kamran Alipour, Jurgen P Schulze, Yi Yao, Avi Ziskind, and Giedrius Burachas. 2020. A study on multimodal and interactive explanations for visual question answering. *arXiv preprint arXiv:2003.00431* (2020).

[4] Ariful Islam Anik and Andrea Bunt. 2021. Data-centric explanations: explaining training data of machine learning systems to promote transparency. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 1–13.

[5] Sule Anjomshoae, Amro Najjar, Davide Calvaresi, and Kary Främling. 2019. Explainable agents and robots: Results from a systematic literature review. In *18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019.* International Foundation for Autonomous Agents and Multiagent Systems, 1078–1088.

[6] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115.

[7] 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.

[8] 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.

[9] 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.

[10] Jaime Banks. 2020. Optimus primed: Media cultivation of robot mental models and social judgments. *Frontiers in Robotics and AI* 7 (2020), 62.

[11] Gagan Bansal, Besmira Nushi, Ece Kamar, Walter S Lasecki, Daniel S Weld, and Eric Horvitz. 2019. Beyond accuracy: The role of mental models in human-AI team performance. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, Vol. 7. 2–11.

[12] Gagan Bansal, Tongshuang Wu, Joyce Zhu, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel S Weld. 2020. Does the whole exceed its parts? the effect of ai explanations on complementary team performance. *arXiv preprint arXiv:2006.14779* (2020).

[13] Cristian-Paul Bara, Sky CH-Wang, and Joyce Chai. 2021. MindCraft: Theory of mind modeling for situated dialogue in collaborative tasks. *arXiv preprint arXiv:2109.06275* (2021).

[14] Matthias Beggiato and Josef F Krems. 2013. The evolution of mental model, trust and acceptance of adaptive cruise control in relation to initial information. *Transportation research part F: traffic psychology and behaviour* 18 (2013), 47–57.

[15] Matthias Beggiato, Marta Pereira, Tibor Petzoldt, and Josef Krems. 2015. Learning and development of trust, acceptance and the mental model of ACC. A longitudinal on-road study. *Transportation research part F: traffic psychology and behaviour* 35 (2015), 75–84.

[16] Erin Beneteau, Olivia K Richards, Mingrui Zhang, Julie A Kientz, Jason Yip, and Alexis Hiniker. 2019. Communication breakdowns between families and Alexa. In *Proceedings of the 2019 CHI conference on human factors in computing systems.* 1–13.

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

[18] Umang Bhatt, Javier Antorán, Yunfeng Zhang, Q Vera Liao, Prasanna Sattigeri, Riccardo Fogliato, Gabrielle Melançon, Ranganath Krishnan, Jason Stanley, Omesh Tickoo, et al. 2021. Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society.* 401–413.

[19] Clara Bove, Jonathan Aigrain, Marie-Jeanne Lesot, Charles Tijus, and Marcin Detyniecki. 2022. Contextualization and Exploration of Local Feature Importance Explanations to Improve Understanding and Satisfaction of Non-Expert Users. In *27th International Conference on Intelligent User Interfaces.* 807–819.

[20] Virginia Braun and Victoria Clarke. 2012. Thematic analysis. (2012).

[21] Zana Buçinca, Phoebe Lin, Krzysztof Z Gajos, and Elena L Glassman. 2020. Proxy tasks and subjective measures can be misleading in evaluating explainable ai systems. In *Proceedings of the 25th International Conference on Intelligent User Interfaces.* 454–464.

[22] Andrea Bunt, Matthew Lount, and Catherine Lauzon. 2012. Are explanations always important? A study of deployed, low-cost intelligent interactive systems. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces.* 169–178.

[23] Tathagata Chakraborti, Anagha Kulkarni, Sarath Sreedharan, David E Smith, and Subbarao Kambhampati. 2019. Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior. In *Proceedings of the international conference on automated planning and scheduling*, Vol. 29. 86–96.

[24] Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. 2020. The emerging landscape of explainable ai planning and decision making. *arXiv preprint arXiv:2002.11697* (2020).

[25] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. *arXiv preprint arXiv:1701.08317* (2017).

[26] Li Chen and Feng Wang. 2017. Explaining recommendations based on feature sentiments in product reviews. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces.* 17–28.

[27] Janghee Cho. 2018. Mental models and home virtual assistants (HVAs). In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems.* 1–6.

[28] Michael Chromik, Malin Eiband, Felicitas Buchner, Adrian Krüger, and Andreas Butz. 2021. I think i get your point, AI! the illusion of explanatory depth in explainable AI. In *26th International Conference on Intelligent User Interfaces.* 307–317.

[29] Eric Corbett and Astrid Weber. 2016. What can I say? addressing user experience challenges of a mobile voice user interface for accessibility. In *Proceedings of the 18th international conference on human-computer interaction with mobile devices and services.* 72–82.

[30] Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. 2020. A survey of the state of explainable AI for natural language processing. *arXiv preprint arXiv:2010.00711* (2020).

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[32] 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.

[33] Jonathan Dodge, Q Vera Liao, Yunfeng Zhang, Rachel KE Bellamy, and Casey Dugan. 2019. Explaining models: an empirical study of how explanations impact fairness judgment. In *Proceedings of the 24th international conference on intelligent user interfaces.* 275–285.

[34] Tim Donkers, Timm Kleemann, and Jürgen Ziegler. 2020. Explaining recommendations by means of aspect-based transparent memories. In *Proceedings of the 25th International Conference on Intelligent User Interfaces.* 166–176.

[35] Kate Ehrlich, Susanna E Kirk, John Patterson, Jamie C Rasmussen, Steven I Ross, and Daniel M Gruen. 2011. Taking advice from intelligent systems: the double-edged sword of explanations. In *Proceedings of the 16th international conference on Intelligent user interfaces.* 125–134.

[36] Upol Ehsan, Q Vera Liao, Michael Muller, Mark O Riedl, and Justin D Weisz. 2021. Expanding explainability: towards social transparency in AI systems. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 1–19.

[37] Upol Ehsan and Mark O Riedl. 2020. Human-centered explainable ai: towards a reflective sociotechnical approach. In *International Conference on Human-Computer Interaction.* Springer, 449–466.

[38] Upol Ehsan and Mark O Riedl. 2021. Explainability Pitfalls: Beyond Dark Patterns in Explainable AI. *arXiv preprint arXiv:2109.12480* (2021).

[39] 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.

[40] John Fox, Sanford Weisberg, Daniel Adler, Douglas Bates, Gabriel Baud-Bovy, Steve Ellison, David Firth, Michael Friendly, Gregor Gorjanc, Spencer Graves, et al. 2012. Package 'car'. *Vienna: R Foundation for Statistical Computing* 16 (2012).

[41] Maria Fox, Derek Long, and Daniele Magazzeni. 2017. Explainable planning. *arXiv preprint arXiv:1709.10256* (2017).

[42] 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.

[43] 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.

[44] 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.

[45] Katy Ilonka Gero, Zahra Ashktorab, Casey Dugan, Qian Pan, James Johnson, Werner Geyer, Maria Ruiz, Sarah Miller, David R. Millen, Murray Campbell, Sadhana Kumaravel, and Wei Zhang. 2020. Mental Models of AI Agents in a Cooperative Game Setting. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376316

[46] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, 80–89.

[47] Ileana Maria Greca and Marco Antonio Moreira. 2000. Mental models, conceptual models, and modelling. *International journal of science education* 22, 1 (2000), 1–11.

[48] 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.* 1–11.

[49] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)* 51, 5 (2018), 1–42.

[50] Lijie Guo, Elizabeth M Daly, Oznur Alkan, Massimiliano Mattetti, Owen Cornec, and Bart Knijnenburg. 2022. Building Trust in Interactive Machine Learning via User Contributed Interpretable Rules. In *27th International Conference on Intelligent User Interfaces.* 537–548.

[51] Tihomir Gvero and Viktor Kuncak. 2015. Interactive synthesis using free-form queries. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2. IEEE, 689–692.

[52] Sophia Hadash, Martijn C Willemsen, Chris Snijders, and Wijnand A IJsselsteijn. 2022. Improving understandability of feature contributions in model-agnostic explainable AI tools. In *CHI Conference on Human Factors in Computing Systems.* 1–9.

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

[54] 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.

[55] James Jaccard, Michael A Becker, and Gregory Wood. 1984. Pairwise multiple comparison procedures: A review. *Psychological Bulletin* 96, 3 (1984), 589.

[56] Maia Jacobs, Jeffrey He, Melanie F. Pradier, Barbara Lam, Andrew C Ahn, Thomas H McCoy, Roy H Perlis, Finale Doshi-Velez, and Krzysztof Z Gajos. 2021. Designing AI for trust and collaboration in time-constrained medical decisions: a sociotechnical lens. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 1–14.

[57] 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.

[58] Mohit Jain, Pratyush Kumar, Ramachandra Kota, and Shwetak N Patel. 2018. Evaluating and informing the design of chatbots. In *Proceedings of the 2018 Designing Interactive Systems Conference.* 895–906.

[59] 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.

[60] Jiepu Jiang, Wei Jeng, and Daqing He. 2013. How do users respond to voice input errors? Lexical and phonetic query reformulation in voice search. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval.* 143–152.

[61] 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.

[62] Pigi Kouki, James Schaffer, Jay Pujara, John O'Donovan, and Lise Getoor. 2019. Personalized explanations for hybrid recommender systems. In *Proceedings of the 24th International Conference on Intelligent User Interfaces.* 379–390.

[63] Charlene Krueger and Lili Tian. 2004. A comparison of the general linear mixed model and repeated measures ANOVA using a dataset with multiple missing data points. *Biological research for nursing* 6, 2 (2004), 151–157.

[64] Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. 2012. Tell me more? The effects of mental model soundness on personalizing an intelligent agent. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* 1–10.

[65] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. 2013. Too much, too little, or just right? Ways explanations impact end users' mental models. In *2013 IEEE Symposium on visual languages and human centric computing.* IEEE, 3–10.

[66] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. 2017. Explainable agency for intelligent autonomous systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31. 4762–4763.

[67] Sau-lai Lee, Ivy Yee-man Lau, Sara Kiesler, and Chi-Yue Chiu. 2005. Human mental models of humanoid robots. In *Proceedings of the 2005 IEEE international conference on robotics and automation.* IEEE, 2767–2772.

[68] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip Torr. 2019. Controllable text-to-image generation. *Advances in Neural Information Processing Systems* 32 (2019).

[69] Fei Li and HV Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment* 8, 1 (2014), 73–84.

[70] Hao Li, Yu-Ping Wang, Jie Yin, and Gang Tan. 2019. SmartShell: Automated Shell Scripts Synthesis from Natural Language. *International Journal of Software Engineering and Knowledge Engineering* 29, 02 (2019), 197–220.

[71] 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.

[72] 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.

[73] Q Vera Liao, Daniel Gruen, and Sarah Miller. 2020. Questioning the AI: informing design practices for explainable AI user experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems.* 1–15.

[74] 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.

[75] Q Vera Liao and Kush R Varshney. 2021. Human-Centered Explainable AI (XAI): From Algorithms to User Experiences. *arXiv preprint arXiv:2110.10790* (2021).

[76] Brian Y Lim and Anind K Dey. 2009. Assessing demand for intelligibility in context-aware applications. In *Proceedings of the 11th international conference on Ubiquitous computing*. 195–204.

[77] Gabriel Lima, Nina Grgić-Hlača, and Meeyoung Cha. 2021. Human perceptions on moral responsibility of AI: A case study in AI-assisted bail decision-making. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–17.

[78] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. 2020. Explainable ai: A review of machine learning interpretability methods. *Entropy* 23, 1 (2020), 18.

[79] Jason Linder, Gierad Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, and Eytan Adar. 2013. PixelTone: A multimodal interface for image editing. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. 2829–2830.

[80] Joseph Lindley, Haider Ali Akmal, Franziska Pilling, and Paul Coulton. 2020. Researching AI Legibility through Design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[81] Ryan Louie, Andy Coenen, Cheng Zhi Huang, Michael Terry, and Carrie J Cai. 2020. Novice-AI music co-creation via AI-steering tools for deep generative models. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–13.

[82] Ryan Louie, Any Cohen, Cheng-Zhi Anna Huang, Michael Terry, and Carrie J Cai. 2020. Cococo: AI-Steering Tools for Music Novices Co-Creating with Generative Models.. In *HAI-GEN+ user2agent@ IUI*.

[83] 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.

[84] Siwen Luo, Hamish Ivison, Caren Han, and Josiah Poon. 2021. Local Interpretations for Explainable Natural Language Processing: A Survey. *arXiv preprint arXiv:2103.11072* (2021).

[85] Michael Katz, et al. 2022. AI Planning Service. https://github.com/IBM/AIPlanningService

[86] Martijn Millecamp, Nyi Nyi Htun, Cristina Conati, and Katrien Verbert. 2019. To explain or not to explain: the effects of personal characteristics when explaining music recommendations. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 397–407.

[87] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence* 267 (2019), 1–38.

[88] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. *arXiv preprint arXiv:1802.05695* (2018).

[89] 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

[90] Milda Norkute, Nadja Herger, Leszek Michalak, Andrew Mulder, and Sally Gao. 2021. Towards explainable AI: Assessing the usefulness and impact of added explainability features in legal document summarization. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–7.

[91] Donald A Norman. 2014. Some observations on mental models. In *Mental models*. Psychology Press, 15–22.

[92] Mahsan Nourani, Chiradeep Roy, Jeremy E Block, Donald R Honeycutt, Tahrima Rahman, Eric Ragan, and Vibhav Gogate. 2021. Anchoring Bias Affects Mental Model Formation and User Reliance in Explainable AI Systems. In *26th International Conference on Intelligent User Interfaces*. 340–350.

[93] Jeroen Ooge, Shotallo Kato, and Katrien Verbert. 2022. Explaining Recommendations in E-Learning: Effects on Adolescents' Trust. In *27th International Conference on Intelligent User Interfaces*. 93–105.

[94] Majdi Owda, Zuhair Bandar, and Keeley Crockett. 2007. Conversation-based natural language interface to relational databases. In *2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*. IEEE, 363–367.

[95] 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.

[96] Cecilia Panigutti, Andrea Beretta, Fosca Giannotti, and Dino Pedreschi. 2022. Understanding the impact of explanations on advice-taking: a user study for AI-based clinical Decision Support Systems. In *CHI Conference on Human Factors in Computing Systems*. 1–9.

[97] 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.

[98] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*. 149–157.

[99] Kun Qian, Marina Danilevsky, Yannis Katsis, Ban Kawas, Erick Oduor, Lucian Popa, and Yunyao Li. 2021. XNLP: A Living Survey for XAI Research in Natural Language Processing. In *26th International Conference on Intelligent User Interfaces*. 78–80.

[100] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[101] Justus Robertson, Athanasios Vasileios Kokkinakis, Jonathan Hook, Ben Kirman, Florian Block, Marian F Ursu, Sagarika Patra, Simon Demediuk, Anders Drachen, and Oluseyi Olarewaju. 2021. Wait, but why?: assessing behavior explanation strategies for real-time strategy games. In *26th International Conference on Intelligent User Interfaces*. 32–42.

[102] 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.

[103] Andrew Ross, Nina Chen, Elisa Zhao Hang, Elena L Glassman, and Finale Doshi-Velez. 2021. Evaluating the interpretability of generative models by interactive reconstruction. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[104] Juliano Efson Sales, Siegfried Handschuh, and André Freitas. 2017. Semeval-2017 task 11: end-user development using natural language. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. 556–564.

[105] Maarten PD Schadd, Tjeerd AJ Schoonderwoerd, Karel van den Bosch, Olaf H Visker, Tjalling Haije, and Kim HJ Veltman. 2022. "I'm Afraid I Can't Do That, Dave"; Getting to Know Your Buddies in a Human–Agent Team. *Systems* 10, 1 (2022), 15.

[106] James Schaffer, Prasanna Giridhar, Debra Jones, Tobias Höllerer, Tarek Abdelzaher, and John O'donovan. 2015. Getting the message? A study of explanation interfaces for microblog data analysis. In *Proceedings of the 20th international conference on intelligent user interfaces*. 345–356.

[107] James Schaffer, John O'Donovan, James Michaelis, Adrienne Raglin, and Tobias Höllerer. 2019. I can do better than your AI: expertise and explanations. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 240–251.

[108] Timo Schick and Hinrich Schütze. 2021. Few-shot text generation with natural language instructions. Association for Computational Linguistics.

[109] Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo. 2012. Making hybrid plans more clear to human users-a formal approach for generating sound explanations. In *Twenty-second international conference on automated planning and scheduling*.

[110] 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.

[111] Amit Sharma and Dan Cosley. 2013. Do social explanations work? Studying and modeling the effects of social explanations in recommender systems. In *Proceedings of the 22nd international conference on World Wide Web*. 1133–1144.

[112] Nasrullah Sheikh, Xiao Qin, Berthold Reinwald, Christoph Miksovic, Thomas Gschwind, and Paolo Scotton. 2020. Knowledge Graph Embedding using Graph Convolutional Networks with Relation-Aware Attention. In *The 2nd International Workshop on Deep Learning on Graphs: Methods and Applications (KDD-DLG 2020)*.

[113] Jiho Shin and Jaechang Nam. 2021. A Survey of Automatic Code Generation from Natural Language. *Journal of Information Processing Systems* 17, 3 (2021), 537–555.

[114] 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.

[115] Jiao Sun, Q Vera Liao, Michael Muller, Mayank Agarwal, Stephanie Houde, Kartik Talamadupula, and Justin D Weisz. 2022. Investigating Explainability of Generative AI for Code through Scenario-based Design. In *27th International Conference on Intelligent User Interfaces*. 212–228.

[116] Harini Suresh, Kathleen M Lewis, John Guttag, and Arvind Satyanarayan. 2022. Intuitively assessing ml model reliability through example-based explanations and editing model inputs. In *27th International Conference on Intelligent User Interfaces*. 767–781.

[117] Maxwell Szymanski, Martijn Millecamp, and Katrien Verbert. 2021. Visual, textual or hybrid: the effect of user expertise on different explanations. In *26th International Conference on Intelligent User Interfaces*. 109–119.

[118] Nathan L Tenhundfeld, Hannah M Barr, HO Emily, and Kristin Weger. 2021. Is my Siri the same as your Siri? An exploration of users' mental model of virtual personal assistants, implications for trust. *IEEE Transactions on Human-Machine Systems* 52, 3 (2021), 512–521.

[119] I Tiddi et al. 2020. Foundations of explainable knowledge-enabled systems. *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges* 47 (2020), 23.

[120] Chun-Hua Tsai, Yue You, Xinning Gui, Yubo Kou, and John M Carroll. 2021. Exploring and promoting diagnostic transparency and explainability in online symptom checkers. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–17.

[121] Prasetya Utama, Nathaniel Weir, Fuat Basik, Carsten Binnig, Ugur Çetintemel, 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).

[122] Lina Varonina and Stefan Kopp. 2021. Knowledge Modelling for Establishment of Common Ground in Dialogue Systems. *IJCoL. Italian Journal of Computational Linguistics* 7, 7-1, 2 (2021), 9–26.

[123] Sahil Verma, John Dickerson, and Keegan Hines. 2020. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596* (2020).

[124] Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714* (2019).

[125] Jennifer Villareale and Jichen Zhu. 2021. Understanding Mental Models of AI through Player-AI Interaction. *arXiv preprint arXiv:2103.16168* (2021).

[126] Justin Walden, Eun Hwa Jung, S Shyam Sundar, and Ariel Celeste Johnson. 2015. Mental models of robots among senior citizens: An interview study of interaction expectations and design implications. *Interaction Studies* 16, 1 (2015), 68–88.

[127] Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. 2019. Designing theory-driven user-centric explainable AI. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–15.

[128] Qiaosi Wang, Koustuv Saha, Eric Gregori, David Joyner, and Ashok Goel. 2021. Towards mutual theory of mind in human-ai interaction: How language reflects what students perceive about a virtual teaching assistant. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.

[129] Xinru Wang and Ming Yin. 2021. Are explanations helpful? a comparative study of the effects of explanations in ai-assisted decision-making. In *26th International Conference on Intelligent User Interfaces*. 318–328.

[130] Yunlong Wang, Priyadarshini Venkatesh, and Brian Y Lim. 2022. Interpretable Directed Diversity: Leveraging Model Explanations for Iterative Crowd Ideation. In *CHI Conference on Human Factors in Computing Systems*. 1–28.

[131] 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.

[132] Justin D Weisz, Michael Muller, Stephanie Houde, John Richards, Steven I Ross, Fernando Martinez, Mayank Agarwal, and Kartik Talamadupula. 2021. Perfection not required? Human-AI partnerships in code translation. In *26th International Conference on Intelligent User Interfaces*. 402–412.

[133] Katharina Weitz, Lindsey Vanderlyn, Ngoc Thang Vu, and Elisabeth André. 2021. " It's our fault!": insights into users' understanding and interaction with an explanatory collaborative dialog system. (2021).

[134] 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.

[135] Yao Xie, Melody Chen, David Kao, Ge Gao, and Xiang'Anthony' Chen. 2020. CheXplain: enabling physicians to explore and understand data-driven, AI-enabled medical imaging analysis. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[136] Fumeng Yang, Zhuanyi Huang, Jean Scholtz, and Dustin L Arendt. 2020. How do visual explanations foster end users' appropriate trust in machine learning?. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*. 189–201.

[137] Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696* (2017).

[138] 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.

[139] 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.

[140] Enhao Zhang and Nikola Banovic. 2021. Method for Exploring Generative Adversarial Networks (GANs) via Automatically Generated Image Galleries. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[141] Wencan Zhang and Brian Y Lim. 2022. Towards Relatable Explainable AI with the Perceptual Process. In *CHI Conference on Human Factors in Computing Systems*. 1–24.

[142] Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2020. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics* 46, 1 (2020), 53–93.

[143] Xiaohan Zou. 2020. A survey on application of knowledge graph. In *Journal of Physics: Conference Series*, Vol. 1487. IOP Publishing, 012016.

## A NATURAL LANGUAGE TO FLOW SYSTEM IMPLEMENTATION DETAILS

The natural language to flow system has three steps, as shown in Figure 8: 1) parsing, 2) matching, and 3) ordering. We provide visualizations of the system outputs for parsing and matching for the natural language input: *"When there's a new message on slack, create files on dropbox"*.

### A.1 Parsing

The system first parses the natural language input using the Abstract Meaning Representation (AMR); as we discuss in Section 3.3.1, this allows our parse to be agnostic to syntactic variations in utterances describing the same flow [9]. The graph has a root and is directed. The nodes and edges of the graph are labeled with properties and roles that the corresponding word plays in the sentence, depending on the known concepts that the AMR model has been trained with. Figure 9 shows an AMR parse tree. The system then uses heuristics to detect triples that can represent components (applications, actions, and objects) using the parts of speech, and edge and node types, predicted by AMR.

### A.2 Matching

The matching is done using a knowledge graph. Figure 10 shows a visualization of the knowledge graph for two applications, Slack and Dropbox. The objects for those two applications are connected to their own applications, as well as related applications. For each node, the knowledge graph contains metadata, like descriptions and the types of data that are involved, and the operations that can take place for an object. The system attempts to match the relevant words from the parse in the previous step to the nodes and metadata in the knowledge graph. This results in a set of candidate components for a flow that include an application name, object and operation.

### A.3 Planning

The final planning stage acts as a constraint satisfaction problem with the following objective:

- Enforce that all detected components are in the flow;
- Enforce the a trigger, if detected, makes it to the beginning of the flow;
- Enforce that all detected orderings are maintained, unless overridden by the trigger in case of a conflict; and
- Pipe outputs of one component to the input of another, based on the name of the object property as specified in the API specification of that component – this makes flows that have known data flow among components more likely, in addition to maintaining that the previous constraints hold.
- Produce a list of required properties of the detected components that the user needs to fill in to complete the flow before deployment.

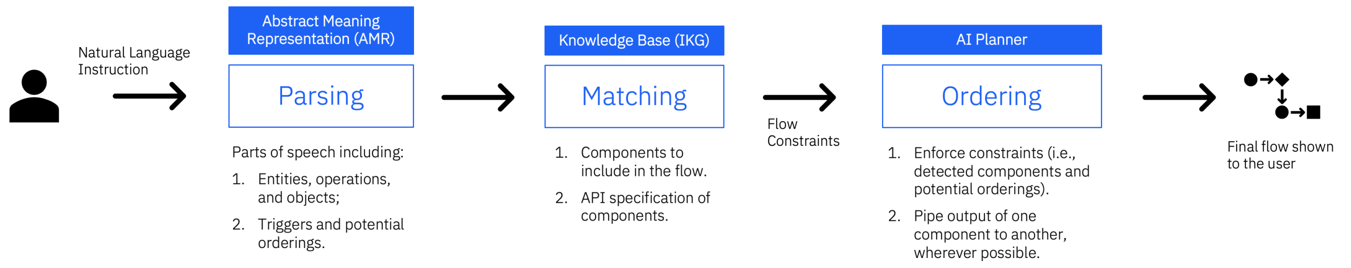

Natural Language Instruction
Abstract Meaning Representation (AMR)
Parsing
Parts of speech including:
1. Entities, operations, and objects;
2. Triggers and potential orderings.
Knowledge Base (IKG)
Matching
1. Components to include in the flow.
2. API specification of components.
Flow Constraints
AI Planner
Ordering
1. Enforce constraints (i.e., detected components and potential orderings).
2. Pipe output of one component to another, wherever possible.
Final flow shown to the user

**Figure 8: The processing pipeline for the natural language to flow system, comprising of three stages of 1) Parsing what the user said, 2) Matching to what the system understands; and 3) Ordering identified components in the desired flow.**

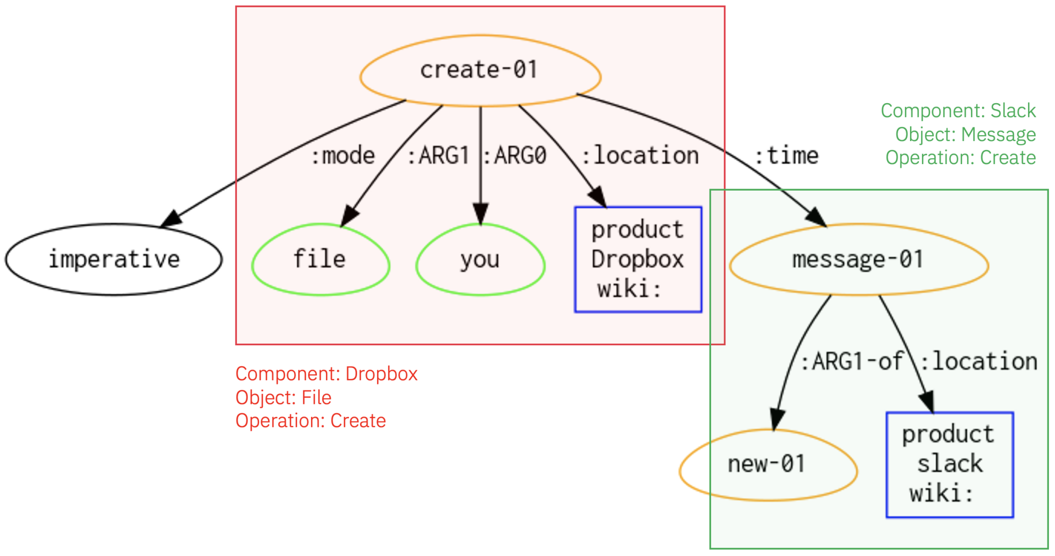


**Figure 9: The AMR representation for the natural language input: *"When there's a new message on slack, create files on dropbox"*. The (red) overlay indicates the transformation of the detected entities later in the Matching stage, they are not part of the AMR representation (notice how the "new message" instruction is interpreted as a CREATE operation even though not explicitly specified.**

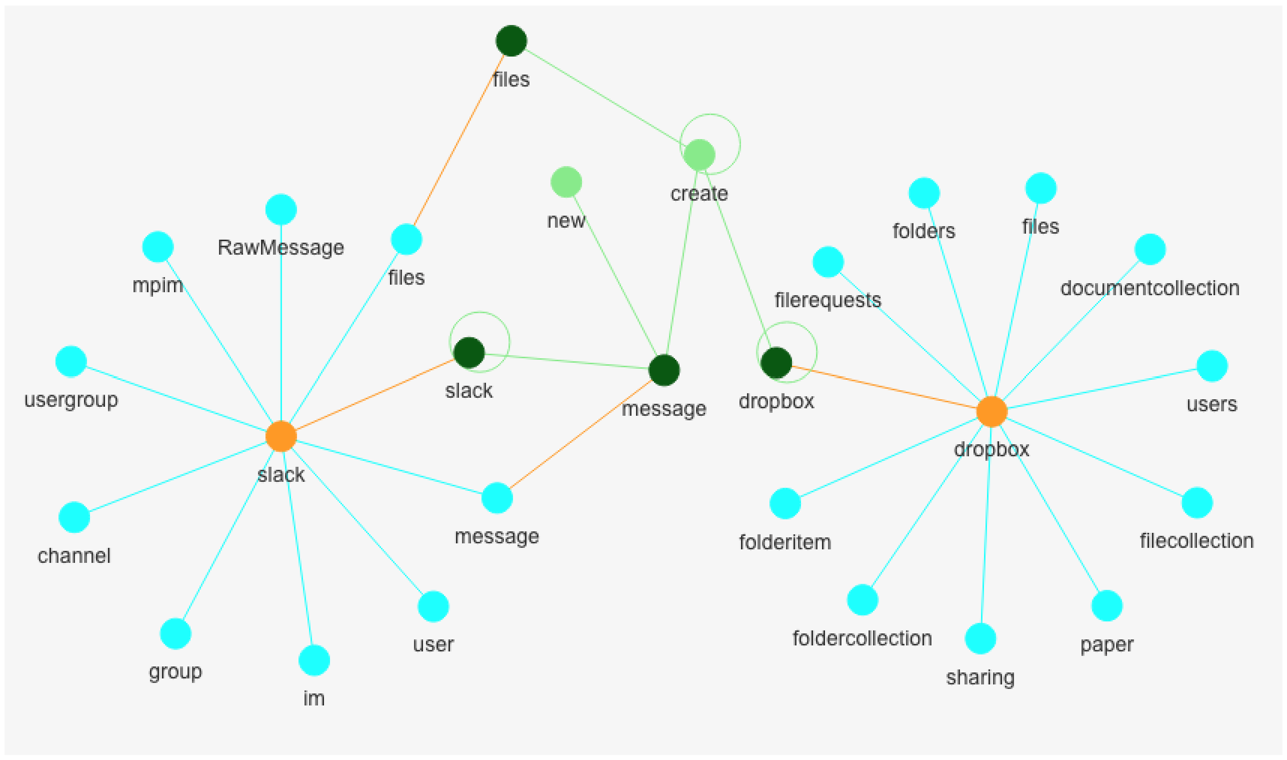


**Figure 10: The knowledge graph representation of the entities detected in the natural language input: *"When there's a new message on slack, create files on dropbox". These are overlaid in red on the AMR parse tree in Figure 9 – this is the purpose of the Matching stage in the pipeline, i.e. to convert the items detected in the user input to entities understood by the system.***