Counterfactual Explanations for Natural Language Interfaces

George Tolkachev
University of Pennsylvania
georgeto@seas.upenn.edu

Stephen Mell
University of Pennsylvania
sm1@seas.upenn.edu

Steve Zdancewic
University of Pennsylvania
stevez@seas.upenn.edu

Osbert Bastani
University of Pennsylvania
obastani@seas.upenn.edu

Abstract

A key challenge facing natural language interfaces is enabling users to understand the capabilities of the underlying system. We propose a novel approach for generating explanations of a natural language interface based on semantic parsing. We focus on counterfactual explanations, which are post-hoc explanations that describe to the user how they could have minimally modified their utterance to achieve their desired goal. In particular, the user provides an utterance along with a demonstration of their desired goal; then, our algorithm synthesizes a paraphrase of their utterance that is guaranteed to achieve their goal. In two user studies, we demonstrate that our approach substantially improves user performance, and that it generates explanations that more closely match the user’s intent compared to two ablations.

1 Introduction

Semantic parsing is a promising technique for enabling natural language user interfaces (Ge and Mooney, 2005; Artzi and Zettlemoyer, 2013; Berant et al., 2013; Wang et al., 2015). However, a key challenge facing semantic parsing is the richness of human language, which can often encode concepts (e.g., “circle”) that do not exist in the underlying system or are encoded using different language (e.g., “ball”). Thus, human users can have trouble providing complex compositional commands in the form of natural language to such systems.

One approach to addressing this issue is to develop increasingly powerful models for understanding natural language (Gardner et al., 2018; Yin and Neubig, 2018). While there has been enormous progress in this direction, there remains a wide gap between what these models are capable of compared to human understanding (Lake and Baroni, 2018), manifesting in the fact that these models can fail in unexpected ways (Ribeiro et al., 2016). This gap can be particularly problematic for end users who do not understand the limitations of machine learning models, since it encourages the human user to provide complex commands, but then performs unreliably on such commands.

Thus, an important problem is to devise techniques for explaining these models. Generally speaking, a range of techniques have recently been developed for explaining machine learning models. The first technique is to use models that are intrinsically explainable, such as linear regression or decision trees. However, this approach also suffers from the fact that showing a decision tree or regression model is likely not useful to an end user.

Instead, we consider an alternative form of explanation called a counterfactual explanation (Wachter et al., 2017). These explanations are designed to describe alternative outcomes to the user. In particular, given a prediction for a specific input, they tell the user how they could have minimally modified that input to achieve a different outcome. As an example, suppose a bank is using a machine learning model to help decide whether to provide a loan to an individual; if that individual is denied the loan, then the bank can provide them with a counterfactual explanation describing how they could change their covariates (e.g., increase their income) to qualify for a loan.

We propose a novel algorithm for computing counterfactual explanations for semantic parsers. In particular, suppose that a user provides a com-
User command 1: “Go to the blue circle”
User command 2: “Go to the top right”
Our explanation: “Go to the blue ball”

Figure 1: Example BabyAI task (from Chevalier-Boisvert et al. (2018)), utterances, and our explanation.

mand in the form of a natural language utterance. If the natural language interface fails to provide the desired result, then our goal is to explain how the user could have modified their utterance to achieve the desired result. To this end, we have the human additionally provide the desired result. Then, we compute an alternative utterance that the semantic parser correctly processes while being as similar as possible to the original utterance. Intuitively, this explanation enables the user to modify their language to reliably achieve their goals in future interactions with the system.

We evaluate our approach on the BabyAI environment (Chevalier-Boisvert et al., 2018), where the human can provide a virtual agent with commands to achieve complex tasks such as “pick up the green ball and place it next to the blue box”. We perform two user studies, which demonstrate that our approach both produces correct explanations (i.e., match the user’s desired intent), and that it substantially improves the user’s ability to provide valid commands.

Example. In Figure 1, we show an example of a BabyAI task along with a user-provided utterance commanding the agent to go to the blue ball. The first command corresponds to a valid program, but cannot be understood by the semantic parser due to the use of the terminology “circle” instead of “ball”. The second command uses the construct “top right” that does not exist in the language. In both cases, the user provides a demonstration where the agent navigates next to the blue ball, upon which our approach generates the explanation shown.

Related work. There has been a great deal of recent interest in providing explanations of black-box machine learning models, focusing on explaining why the model makes an individual prediction (Ribeiro et al., 2016; Lei et al., 2016; Ribeiro et al., 2018; Alvarez-Melis and Jaakkola, 2018; Liu et al., 2018), or achieving better understanding of the limitations of models (Wallace et al., 2019; Ribeiro et al., 2020). In contrast, our goal is to explain how the input can be changed to achieve a desired outcome, which is called a counterfactual explanation (Wachter et al., 2017; Ustun et al., 2019). There has been interest in improving the performance of semantic parsers through interaction (Wang et al., 2016, 2017); our approach is complementary to this line of work, since it aims to make the system more transparent to the user. There has also been work on leveraging natural language descriptions to help generate counterfactual explanations for image classifiers (Hendricks et al., 2018), but not tailored at counterfactual predictions for natural language tasks; specifically, while their approach produces counterfactual explanations in natural language, they are for image predictions rather than text predictions.

For natural language processing tasks, a key challenge is that the input space is discrete (e.g., a natural language utterance); for such settings, there has been work on algorithms for searching over combinatorial spaces of counterfactual explanations (Ross et al., 2021b; Wu et al., 2021; Ross et al., 2021a). However, even for these approaches, the output space is typically small (e.g., a binary sentiment label). In contrast, semantic parsing has highly structured outputs (i.e., programs), requiring significantly different search procedures to find an explanation that produces the correct output. To address this challenge, we define a search space over counterfactual explanations for semantic parsing such that search is tractable.

2 Algorithm

Problem formulation. We consider the problem of computing counterfactual explanations for a semantic parsing model \( f_\theta : \Sigma^* \rightarrow \Pi \). In particular, we assume the user provides a command in the form of an utterance \( s \in \Sigma^* \), with the goal of obtaining some denotation \( y \in \mathcal{Y} \). To achieve the user’s goal, the semantic parsing model produces a program \( \pi = f_\theta(s) \in \Pi \), and then executes the program to obtain denotation \( y = [\pi] \in \mathcal{Y} \), where \([\cdot] : \Pi \rightarrow \mathcal{Y} \) (called the semantics of \( \Pi \)) maps programs to outputs.
In this context, our goal is to provide explanations to the user to help them understand what utterances can be correctly understood and executed by the underlying system. In particular, we assume the user has provided an utterance $s_0$, but the output $[f_\theta(s_0)]$ is not the one that they desired. Then, we ask the user to provide their desired output, after which we provide them with an alternative utterance $s^*$ that is semantically similar to $s_0$ but successfully achieves $y_0$. Formally:

**Definition 2.1.** Given an utterance $s_0 \in \Sigma^*$ and a desired output $y_0 \in \mathcal{Y}$, the counterfactual explanation for $s_0$ and $y_0$ is the sentence

$$ s^* = \arg\min_{s \in L} d(s, s_0) \quad \text{subj. to} \quad [f_\theta(s)] = y_0,$$

where $d$ is a semantic similarity metric and $L \subseteq \Sigma^*$ is the search space of possible explanations.

The goal is that examining $s^*$ should help the user provide utterances that are more likely to be correctly processed in future interactions.

**Search space of explanations.** A key challenge in generating natural language expressions is how to generate expressions that appear natural to the human user. To ensure that our explanations are natural, we restrict to sentences generated by a context-free grammar (CFG) $C$. In particular, we consider explanations in the form of sentences $s \in L(C) \subseteq \Sigma^*$ (where $\Sigma$ is the vocabulary and $L(C)$ is the language generated by $C$). We restrict to sentences with parse trees of bounded depth $d$ in $C$; we denote this subset by $L_d(C)$. In addition, we assume sentences $s \in L_d(C)$ are included in the dataset used to train the semantic parser $f_\theta$ to ensure it correctly parses these sentences.

**Semantic similarity.** Our goal is to compute a sentence $s \in L_d(C)$ that is semantically similar to the user-provided utterance $s_0$. To capture this notion of semantic similarity, we use a pretrained language model $x = g_\theta(s)$ that maps a given sentence $s$ to a vector embedding $x \in \mathbb{R}^K$. Then, we use cosine similarity in this embedding space to measure semantic similarity. In particular, we use the distance $d(s, s_0) = 1 - \text{sim}(g_\theta(s), g_\theta(s_0))$, where $\text{sim}(x, x')$ is the cosine similarity.

**Goal constraint.** Finally, we want to ensure that the provided explanation successfully evaluates to the user’s desired denotation $y_0$. For a given utterance $s$, we can check this constraint simply by evaluating $y = [f_\theta(s)]$ and checking if $y = y_0$.

**Algorithm 1** Our algorithm for computing counterfactual explanations for a semantic parser $f_\theta$.

**procedure** EXPLAIN($s_0, y_0$)

$(s^*, c^*) \leftarrow (\emptyset, -\infty)$

for $s \in L_d(C)$ do

if $[f_\theta(s)] = y_0$ then

$c \leftarrow \text{sim}(g_\theta(s), g_\theta(s_0))$

if $c > c^*$ then $s^*, c^* \leftarrow s, c$

end if

end for

return $s^*$

**end procedure**

**Overall algorithm.** Given user-provided utterance $s_0$ and desired denotation $y_0$, the counterfactual explanation problem is equivalent to:

$$ s^* = \arg\max_{s \in L_d(C)} \sim(g_\theta(s), g_\theta(s_0)) $$

subj. to $[f_\theta(s)] = y_0$.

Assuming $L_d(C)$ is sufficiently small, we can solve this problem by enumerating through the possible choices $s \in L_d(C)$ and choosing the highest scoring one that satisfies the constraint. In practice, we may be able to exploit the structure of the constraint to prune the search space. Our approach is summarized in Algorithm 1.

3 Experiments

We perform two user studies to demonstrate (i) correctness: our explanations preserve the user’s original intent, and (ii) usefulness: our explanations improve user performance.

3.1 BabyAI Task

We evaluate our approach on BabyAI (Chevalier-Boisvert et al., 2018) adapted to our setting. In this task, the human can provide commands to an agent navigating a maze of rooms containing keys, boxes, and balls. The goal is defined by the combination of the agent position and the environment state (e.g., the agent may need to place a ball next to a box). Atomic commands (e.g., going to, picking up, or putting down an object) can then be composed in sequence to achieve complex goals. In our setup, $s_0$ is a natural language command, and $y_0$ is a demonstration in the form of a trajectory the agent could take to achieve the desired goal.

This task comes with a context-free grammar of natural language commands, which we use as the
space of possible explanations. Next, we train a semantic parser to understand commands from this grammar. Since utterances in this grammar correspond one-to-one with programs, we can generate training data. We generate 1000 training examples \((s, \pi)\) consisting of an utterance \(s\) along with a program \(\pi\), and train TranX \((\text{Yin and Neubig}, 2018)\) to predict \(\pi = f_\theta(s)\). For semantic similarity, we use a pretrained DistilBERT model \(q_\theta\) \((\text{Devlin et al., 2018; Sanh et al., 2019})\) to embed utterances \(s\).

Handling the goal constraint is more challenging, since the denotation can be nondeterministic—in particular, multiple different trajectories can be used to achieve a single goal (e.g., there are multiple paths the agent can take to a given object). Thus, if we naively take the denotation of a program to be a single trajectory that achieves the goal, then this trajectory may be different than the given demonstration even if the demonstration also achieves the goal. To address this issue, we instead enumerate the set \(\Pi_y\) of all possible programs that are consistent with the given demonstration \(y\), up to a bounded depth (selected so that \(\Pi_y\) is large enough while ensuring that the experiments still run quickly). Then, we replace the constraint \([f_\theta(s)] = y_0\) with a constraint saying that \(f_\theta(s)\) is in this set—i.e., \(f_\theta(s) \in \Pi_{y_0}\).

### 3.2 Correctness of Explanations

We evaluate whether our explanations are valid paraphrases of the user’s original command.

**Baselines.** We compare to two ablations of our algorithm. The first one omits the goal constraint \(f_\theta(s) \in \Pi_y\); thus, it simply returns the explanation that is most semantically similar to the user-provided utterance \(s_0\). Intuitively, this ablation evaluates the usefulness of the goal constraint.

The second ablation ignores \(s_0\), and returns an explanation \(s\) such that \(f_\theta(s) \in \Pi_{y_0}\); we choose \(s\) to minimize perplexity according to GPT-2. Intuitively, this ablation measures the usefulness of specializing the explanation to the user’s utterance.

**Setup.** We selected 17 BabyAI tasks by randomly sampling BabyAI levels until we obtain a set of tasks of varying difficulty. For example, Task 1 has the simple goal “go to the green ball”, while Task 10 has the more complex goal “pick up a green key, then put the yellow box next to the grey ball”.

Then, our experiment proceeds in two phases. In the first phase, we use Amazon Mechanical Turk (AMT) to collect natural language commands for the agent. For each of our 17 tasks, we show the user a video of the BabyAI agent achieving the task, and then ask them to provide a single command that encodes the goal. In total, we obtained 127 commands (one per user) for each task. Next, for each user instruction, we find the counterfactual explanation according to our algorithm and the two ablations described above.

In the second phase, we conduct a second AMT study to evaluate the correctness of these explanations. In particular, for each of our 17 tasks, we show each participant a single command for that task (chosen randomly from the 127 commands in the first phase), along with the three generated explanations and the video of the agent achieving that task. Then, we ask the user to choose the explanation that is closest in meaning to the original command. We obtained 50 responses.

**Results.** In Table 1, we show the fraction of times users in the second phase selected each explanation, averaged across both users and tasks. Our approach significantly outperforms GPT-2, which is unsurprising since this ablation makes no effort to preserve the user’s intent. Our approach also outperforms the ablation without the goal constraint, demonstrating the usefulness of this constraint.

### 3.3 Usefulness of Explanations

Next, we evaluate whether providing explanations can make it easier for users to provide commands that can be understood by our semantic parser.

**Baselines.** In addition to the two ablations in Section 3.2, we also compare to a baseline where the user is not provided with any explanation.

**Setup.** We run an AMT study similar to the first phase of our study in Section 3.2, except immediately after providing a command for a task, each user is shown an explanation for their command and that task. We collected 50 user responses.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Correctness</th>
<th>Usefulness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>41.4 ± 1.48%</td>
<td>50.8 ± 2.22%</td>
</tr>
<tr>
<td>No demo</td>
<td>34.0 ± 1.42%</td>
<td>49.2 ± 2.01%</td>
</tr>
<tr>
<td>GPT-2</td>
<td>26.4 ± 1.23%</td>
<td>46.2 ± 1.96%</td>
</tr>
<tr>
<td>No training</td>
<td>–</td>
<td>10.2 ± 0.74%</td>
</tr>
</tbody>
</table>

Table 1: Correctness: The frequency at which users chose the explanation generated using the corresponding approach as the best match. Usefulness: The percentage of user utterances correctly parsed (averaged across the last 10 tasks), where users are given explanations generated by the corresponding approach.
Results. For each user command $s_0$, we run our semantic parser to obtain the corresponding program and check whether it is in the set of programs valid for that task—i.e., whether $f_\theta(s_0) \in \Pi_{s_0}$. Table 1 shows the success rate across all users and the last 10 tasks; we restrict to the last 10 to give the user time to learn to improve their performance. Users not provided any explanations performed very poorly overall. The remaining approaches performed similarly; our explanations led to the best performance, followed closely by the ablation without the demonstration, with a wider gap to the ablation that ignores the user utterance. Thus, personalizing the explanation to the user based on their utterance helps improve performance.

4 Conclusion

We have proposed a technique for explaining how users can adapt their utterances to interact with a natural language interface. Our experiments demonstrate how our explanations can be used to significantly improve the usability of semantic parsers when they are limited in terms of their semantic understanding. While any explanations are already very useful, we show that personalizing explanations can further improve performance.

A key design choice in our approach is to construct a synthetic grammar from which counterfactual explanations are generated. In a realistic application, the semantic parsing model can be trained on a combination of synthetic data and real-world data, enabling our approach to be used in conjunction with the synthetic grammar. A key direction for future work is extending our approach to settings where such a grammar is not available. In our experience, a key challenge in this setting is that the generated text can be unnatural, possibly due to the constraints imposed on the search space.

Acknowledgments

We gratefully acknowledge support from NSF CCF-1917852 and CCF-1910769. The views expressed are those of the authors and do not reflect the official policy or position of the U.S. Government.

References


Alexis Ross, Ana Marasović, and Matthew E Peters. 2021b. Explaining nlp models via minimal contrastive editing (mice). In Findings of the ACL.


