The Whole Truth and Nothing But the Truth: Faithful and Controllable Dialogue Response Generation with Dataflow Transduction and Constrained Decoding

Anonymous ACL submission

Abstract

In a real-world dialogue system, generated text must satisfy several interlocking constraints: informativeness, truthfulness, and ease of control. The two predominant paradigms in language generation—neural language modeling and rule-based generation—struggle to satisfy these constraints simultaneously. We describe a hybrid architecture for dialogue response generation that combines the strengths of both paradigms. The first component of this architecture is a rule-based content selection model defined using a new formal framework called *dataflow transduction*, which uses declarative rules to transduce a dialogue agent's actions and their results (represented as dataflow graphs) into context-free grammars representing the space of contextually acceptable responses. The second component is a constrained decoding procedure that uses these grammars to constrain the output of a neural language model, which selects fluent utterances. Our experiments show that this system outperforms both rule-based and learned approaches in human evaluations of fluency, relevance, and truthfulness.

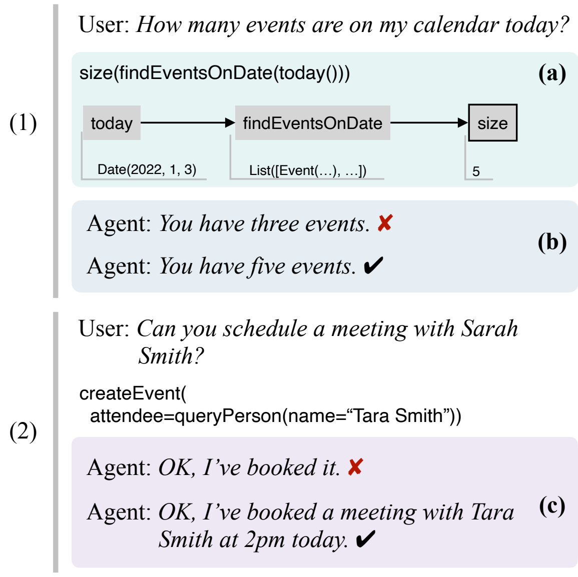

Figure 1: Interaction between a user and a dialogue agent. Once the user's request is translated into an agent action—expressible as a program or dataflow graph (a)—the agent must generate a response. Agent responses might simply state the result of the agent's action, but must do so truthfully (b). Often responses should describe both the action and the result, *e.g.*, to help users identify when the agent has misunderstood their request (c). These responses should be straightforward for system designers to inspect and modify.

1 Introduction

In a task-oriented dialogue system, response generation is naturally posed as a conditional language modeling problem: dialogue agents must produce a contextually appropriate natural language string conditioned on the history of the user and agent interaction. But unlike many language generation problems, a good dialogue response generation model is not (just) a model of typical human utterances in context. Instead, effective dialogue agents must balance fluent generation with a set of much stricter constraints.

Consider the dialogue shown in Fig. 1. In the first turn of this dialogue, the user makes a request, the dialogue agent correctly translates it into a computation—here represented as a dataflow graph (Fig. 1a)—then it needs to accurately describe this computation's return value (Fig. 1b), rather than using an arbitrary number (*e.g.*, 3 in the `Date`) on the dataflow graph. In the second step, the agent may also make a mistake: perhaps because of a speech recognition error, it creates a meeting with *Tara Smith* rather than *Sarah Smith*. Simply describing the result of its action might cause a user to incorrectly conclude that their request was completed successfully. To avoid confusion, a system designer might wish to ensure that the agent instead echoes back to the user the details of the agent's action (Fig. 1c). This example highlights the challenges central to building real-world dialogue response generation systems.

First, response generation is not simply a problem of describing the *result* of a computation in natural language. In some cases, response generators may also usefully **describe the provenance** of that result—the computation itself and its intermediate values. In many human-to-human conversations, a response as detailed as Fig. 1c would be over-informative, violating Grice's maxim of quantity (1975). But for a speaker that is prone to mistakes, such as an AI agent, describing its own understanding can increase user trust when the understanding is accurate and provides an opportunity for correction when it is not.

Second, dialogue response generation systems must **guarantee truthfulness**: as typically the primary source of information about the action that a dialogue agent took, a response generator that describes even a small fraction of these computations incorrectly can produce disastrous results. Importantly, truthful utterances might be low-probability under a domain-general language model (LM), particularly when they reflect errors in language understanding (as in Fig. 1b).

Finally, response generation systems must **support declarative specification of agent behavior**. When confusing or infelicitious responses are discovered, it should be possible to easily and precisely modify them without changing the dialogue agent's behavior in other contexts.

In recent years, the main focus of academic dialogue research has been on "end-to-end" learned models for response generation, especially neural sequence models (Vinyals and Le, 2015; Zhang et al., 2020b). But while such models excel at producing fluent and coherent output, research continues to find that they struggle in maintaining faithfulness (Wiseman et al., 2017; Maynez et al., 2020). Perhaps more fundamentally, because the behavior of such systems is encoded implicitly in their training data, designing a dialogue system requires system builders to write and edit a large number of training examples whose final effect may be difficult to predict.

As a result, many dialogue systems in the real world remain rule-based: system builders handwrite rules (*e.g.*, in the form of a synchronous grammar) for transforming dialogue states into text, and these rules are applied directly during deployment. But such rule-based systems are also notoriously difficult to build and maintain (Walker et al., 2002; Reiter, 2022). They require designers to anticipate every low-level question about surface realization, and to encode these in the same grammar that is responsible for enforcing high-level properties like truthfulness.

Given the many strengths of modern LMs, is there a way to leverage them while satisfying the numerous other demands on dialogue response generation systems? In this paper, we describe a hybrid approach that combines the advantages of end-to-end and rule-based approaches. This approach has two components:

- A dataflow transduction procedure, based on a new formalism that uses declarative rules to map a computation (represented as a dataflow graph) into a context-free grammar (CFG) that defines the space of all responses allowed for the given computation. This formal framework makes it possible to write rules to precisely and truthfully describe both data and its provenance, while performing supplementary computation where needed to produce informative responses.

- A constrained decoding procedure that intersects a CFG with a neural LM, making it possible to decompose language generation into a **content selection model** (implemented by the grammar) and a separate **fluency model** (implemented by an LM).

Together, dataflow transduction and constrained decoding make it possible to build a faithful generation system capable of describing a complex, open-ended, space of tasks. Using a subset of SMCalFlow dialogues (Semantic Machines et al., 2020) and only 187 declarative rules, our hybrid system is consistently rated as more truthful, relevant, and fluent than either a rule-based or end-to-end neural system. Similar results are observed on MultiWOZ dialogues (Budzianowski et al., 2018; Eric et al., 2020). Code, data, and trained models used in our experiments will be released.

2 Problem Formulation

We study the problem of response generation for task-oriented dialogue. A dialogue, like the one in Fig. 1, consists of a sequence of **turns**, each consisting of a **user utterance** x_i, one or more **actions** a_i, and an **agent response** y_i. The job of a **dialogue agent** is to predict an appropriate action and response from a dialogue history, i.e., mapping from $(x_1, a_1, y_1, x_2, a_2, y_2, \ldots, x_n) \mapsto (a_n, y_n)$.

A common approach to building dialogue agents decomposes this prediction process into several steps. First, a **language understanding module** maps from a user utterance (and possibly other components of the dialogue history) to a meaning representation (*e.g.*, a structured user intent, API request or executable program). This meaning representation is evaluated, producing actions a, which are passed to a **response generation module** that produces an agent utterance y.

The focus of this paper is the response generator. We assume that we have a pre-specified language understanding module that maps from conversation histories to **computations**, in the form of short programs, which are then executed to produce actions a. As described by Semantic Machines et al. (2020), these computations may equivalently be viewed as **dataflow graphs** in which each node is labeled with a function, constructor, or primitive value, as well as a return value once the node is executed. We additionally assume access to a dataset of dialogues containing gold-standard user and agent utterances. Given a language understanding module and a dataset of dialogues, we aim to implement a response generator that, when applied to a dataflow graph, satisfies the three properties outlined in §1: description of data and its provenance, guaranteed truthfulness, and declarative specification.

Our response generation system is built from two pieces: (1) a procedure for transducing dataflow graphs into CFGs (§3), and (2) a constrained decoding procedure for intersecting a CFG with a neural LM (§4). Hybrid generation systems of this kind have a long history in NLP (Langkilde and Knight, 1998). Our aim in this paper is to show the benefits of a new generation paradigm based on dataflow transduction, and offer new rule-writing formalisms and decoding algorithms tailored to modern language models.

3 Dataflow Transduction

Given a dataflow graph G (*e.g.*, Fig. 1a) rooted at a node $v_{\texttt{root}}$ (the return value of the program represented by the dataflow graph), our task is to generate a string that describes $v_{\texttt{root}}$ and its provenance. To achieve this, we propose a new formal framework for generation based on **dataflow transduction**. At a high level, the formalism uses declarative rules that describe how to transform a dataflow graph into a graph-specific grammar (specifically a **quasi-synchronous context-free grammar**, or QCFG) that defines the space of allowed responses. These rules walk along the graph, introduce new computations (dataflow subgraphs) as needed, and add rules to the grammar.

Head: S

Body:

```
match computation:
  case findEventsOnDate(date):
    num = size(computation)
    event = head(computation)
    return {"num": num, "event": event, "date": date}
```

Response Template:

I found {LEX <num>} *event* {PP <date>}. *It's* {EVENT <event>}.

Figure 2: A dataflow transduction rule with head S, a body (expressed in Python), and a response template (which queries the dictionary returned by the body).

Formally, a dataflow transducer $\mathcal{S}$ is defined by a 4-tuple $(\mathcal{T}, \Sigma, \mathcal{R}, t_{\texttt{start}})$ where $\mathcal{T}$ is a set of nonterminal types,[1] Σ is the set of terminals (word types), $\mathcal{R}$ is a set of dataflow transduction rules (see §3.1), and $t_{\texttt{start}}$ is the nonterminal type of the start symbol. When applied to G the dataflow transducer expands the graph, yielding a new graph $\bar{G}$, and produces a QCFG.

A QCFG (Smith and Eisner, 2006) is a specialized CFG whose nonterminals include alignments to the nodes $V(\bar{G})$ of $\bar{G}$. Where an ordinary CFG might specify ways to generate an `NP` (noun phrase) or a `DATE`, a QCFG would specify ways to generate an `NP` or `DATE` that describes the result and provenance of v, for each appropriately typed node $v \in V(\bar{G})$. A QCFG resulting from dataflow transduction is a 4-tuple $(\mathcal{T} \times V(\bar{G}), \Sigma, \mathcal{P}, t_{\texttt{start}})$ where $\mathcal{T} \times V(\bar{G})$ is the QCFG's set of nonterminals and $\mathcal{P}$ is its set of productions. A QCFG production has the form $\alpha \to \beta_1\beta_2\cdots\beta_N$ where the left-hand-side $\alpha = (t, v) \in \mathcal{T} \times V(\bar{G})$ is a QCFG nonterminal, and each β_i can be either a nonterminal (t_i, v_i) or a terminal in Σ. The v_i of a right-hand-side nonterminal β_i may have appeared in the original G, or may have been added to $\bar{G}$ by the dataflow transducer. These production rules then derive a set of strings as in an ordinary CFG.

3.1 Dataflow Transduction Rules

A dataflow transduction rule is applied to a node $v \in \bar{G}$ (if v has appropriate properties) to create a

[1] In practice, nonterminal types might correspond to dialogue acts, syntactic categories, semantic categories, etc. This is up to the designer.

single QCFG production $(t, v) \to \cdots$ that could be used to describe v. An example rule is shown in Fig. 2. A rule has three components: (1) a **head**, namely the nonterminal type $t \in \mathcal{T}$; (2) a **body**, which is a piece of code that determines whether the rule can apply to v, and which may look up or create nodes that are related to v; and (3) a **response template**, which specifies the right-hand side of the QCFG production in terms of the related nodes that identified in the body.

Rule Head. This nonterminal type characterizes the type of node that the transduction rule is able to describe and the type of description that it will produce.[1] When a rule with head t is successfully applied to the node v, the resulting QCFG production has left-hand-side (t, v).

Rule Body. A rule body declares the condition when the rule can be applied by examining the dataflow graph $\bar{G}_v$ rooted at v. It can contain executable logic that identifies additional computation nodes that will be recursively described.[2] For example, the rule body in Fig. 2 checks whether $\bar{G}_v$ has the form `findEventsOnDate(date)`. If so, it binds the variable `date` accordingly, and introduces new nodes into $\bar{G}$, bound to the variables `num` and `event`, which compute the number of events and the first event. All three of these variables will be referenced in the response template.

Response Template. The response template says how to create the right-hand side of the QCFG rule—a sequence $\beta_1 \cdots \beta_N$ of terminals and nonterminals. Each QCFG nonterminal $\beta_i = (t_i, v_i)$ specifies a related node $v_i \in V(\bar{G})$ to describe, along with a dataflow nonterminal t_i that says *how* to describe it. The possible descriptions of v_i will thus emerge from applying transducer rules with head t_i to node v_i. In our template syntax, the notation `{EVENT <event>}` would construct the QCFG nonterminal $(\texttt{EVENT}, v)$, if the rule body has bound the variable `event` to the node v. This syntax is illustrated in Fig. 2; *e.g.*, the response template will construct three QCFG nonterminals, with types `LEX`, `PP`, and `EVENT`.

3.2 Dataflow Transduction Procedure

Given a dataflow transducer $\mathcal{S}$ and a dataflow graph G rooted at node $v_{\texttt{root}}$, we can transduce the graph into a QCFG as follows. The system starts out by creating QCFG productions that can expand the start nonterminal $(t_{\texttt{start}}, v_{\texttt{root}})$. For each transduction rule in $\mathcal{R}$ whose head is $t_{\texttt{start}}$, it executes the body, which checks any additional conditions for whether the rule can be applied to $v_{\texttt{root}}$, binds variables, and uses the response template to create a QCFG production. If these productions mention new nonterminals, the system recursively creates further QCFG productions, in the same way, that can expand those nonterminals. As a special case, to expand a nonterminal of the form $(\texttt{LEX}, v)$, the system creates a QCFG production whose right-hand side gives the value of v, as rendered into natural language using a lexicalization function rather than a template; *e.g.*, a value `Long(1)` would be rendered as *"1"*.

The recursive process continues until productions have been created for every nonterminal that appears in the QCFG. The resulting QCFG compactly represents a combinatorial space of possible responses. It will generally include multiple productions aligned to the same node v, created by different dataflow transduction rules.

This mechanism can be used to copy simple values like strings and numbers from the dataflow graph, as well as to create more complex recursive descriptions. Note that (1) transduction rules are selected via their head but also condition on the dataflow graph through their body, and (2) all QCFG nonterminals are grounded in the dataflow graph. Together, this provides a means to ensure truthfulness when generating responses.

4 Constrained Decoding

In this section, we describe how to integrate the formal framework above with a general LM to perform response generation, as illustrated in Fig. 3. Given a derived QCFG of the kind described in §3.2, we perform constrained decoding as in (Shin et al., 2021; Roy et al., 2022), generating response candidates from a pretrained LM.

The QCFG resulting from dataflow transduction implicitly represents a set of possible derivation trees and the agent responses they yield. As long as transduction rules faithfully describe the nodes they apply to, every derivation in this set will correspond to a truthful agent utterance. But these utterances may not always be grammatical or natural. For example, the response template in Fig. 2 may be realized as *"I found 2 event on Monday"* since the rule body does not check whether the value of `num` is 1. Similarly, the response template

[2]Note that the nodes added by the body may represent further computations on existing nodes of $\bar{G}_v$ or may be completely disjoint from the existing nodes.

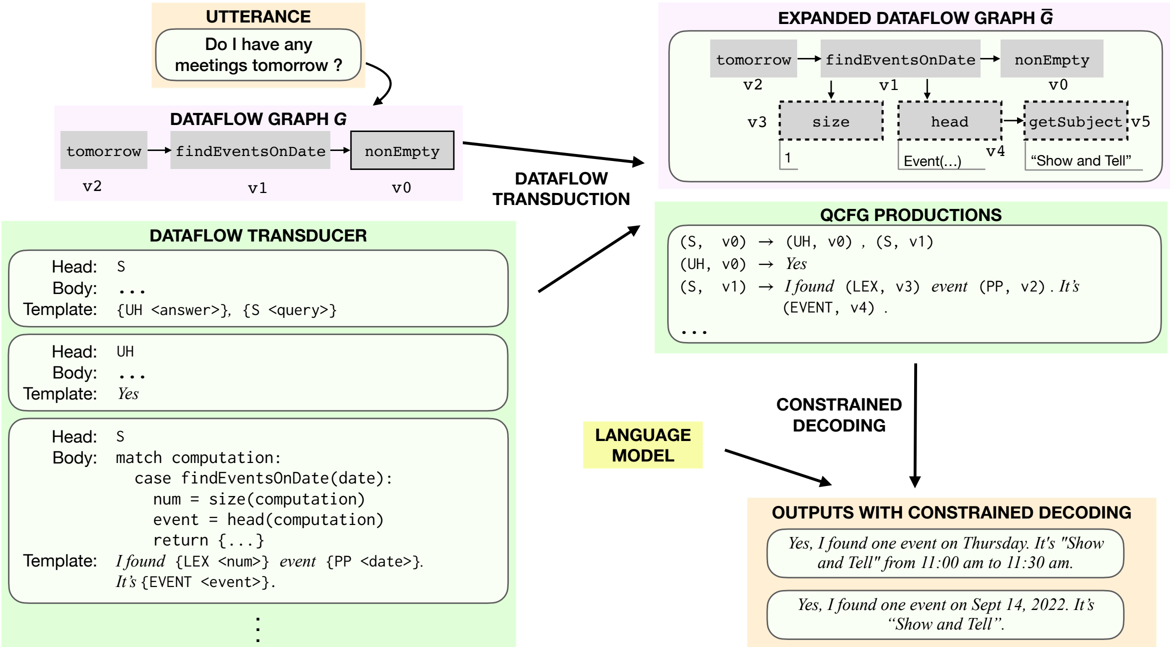

Figure 3: The hybrid response generation approach using dataflow transduction and constrained decoding. Given a computation `nonEmpty(findEventsOnDate(tomorrow()))` for the user utterance *"Do I have any meetings tomorrow"*, we first derive QCFG productions by applying the dataflow transducer to the dataflow graph G using the procedure described in §3.2. This procedure also expands the dataflow graph into $\bar{G}$: for example, the nodes `v3` and `v4` were added by the third transducer rule. Then we extract candidate responses from a LM, constrained by the QCFG. The varying descriptions of the date `v2` and the event `v4` are permitted because the QCFG offers a choice of productions that can be used to expand the (`PP`, `v2`) and (`EVENT`, `v4`) nonterminals. (Those productions and the transducer rules that created them are not shown in the figure. The nodes added by those transducer rules and used by those productions are also not shown, except for `v5`.)

$$\{\texttt{EVENT}\ \langle \textit{event} \rangle\} \text{ starts on } \{\texttt{DATE}\ \langle \textit{date} \rangle\}.$$

may be realized as *The product meeting on Monday starts on Monday*, if the grammar permits identifying events by their dates. With carefully engineered and highly specialized rules (*e.g.*, using extremely fine-grained nonterminal types), it would be possible to ensure that the responses are always fluent and even that there is always a single possible outcome from the top-down search procedure. However, this would usually require much a more complicated set of rules, which creates a burden for system development and maintenance.

Our proposed approach instead uses a large-scale pretrained LM (preferably fine-tuned) to select among truthful utterances produced by the QCFG.[3] One option is to use the LM to re-rank all strings that can be produced by the QCFG, but that would be very computationally expensive. Instead, we follow Shin et al. (2021) and Roy et al. (2022), who decode sentences from a given LM under the constraint that they must be valid under a given CFG. This constrained decoding method uses Earley's algorithm (1970) to incrementally parse the sentence as it is generated and determine the set of words that could grammatically serve as the next token. In contrast to these prior papers, which used a static CFG, we derive a new CFG each time the dialogue agent needs to generate a response, by applying the dataflow transducer to the current dataflow graph.

5 Experiments

To evaluate this approach, we conducted a set of detailed experiments on the SMCalFlow dataset (Semantic Machines et al., 2020) (§5.1–§5.3), and a brief study on applying our approach to the MultiWOZ dataset (Budzianowski et al., 2018) (§5.4).

5.1 Data and Evaluation Metrics

SMCalFlow is a large-scale task-oriented dialogue dataset, in which each user utterance is annotated with a correct dataflow program (i.e., computa-

[3] Of course, decisions deferred to the LM could be encoded in the grammar instead. While this is rarely necessary to ensure grammaticality or fluency, system designers might choose to encode some *pragmatic* decisions, like how much detail to provide, in the grammar rather than in the LM.

System	Automatic Metrics					Human Evaluation (%)		
	BLEU	ROUGE	BERTSc.	R@1	R@5	Grammatical	Relevant	Truthful
QCFG Random Sampling	.35	.58	.50	.02	.06	62.3†	90.9†	92.3
Unconstrained Decoding	.77	.86	.87	.47	.66	98.7	93.3	82.2†
QCFG-Constrained Decoding	.80	.87	.86	.56	.78	99.0	96.6	91.6
Gold	1.0	1.0	1.0	1.0	1.0	99.0	98.0	92.3

Table 1: Evaluation results on SMCalFlow. Automatic metrics are calculated against the gold responses on the full validation set. Human evaluation is conducted on 297 randomly sampled validation examples. †: Results are significantly worse than the "Gold" system ($p < 10^{-4}$, McNemar's test).

tion) and a "gold" response that would be desirable for the agent to produce.[4] We use the v2.0 release processed by Platanios et al. (2021). We focus on a subset of SMCalFlow involving calendar event queries. This subset contains 8938 training examples and 1041 validation examples. We found that 187 transduction rules, written by some of us in a matter of hours, were sufficient to cover all gold system responses in these examples.[5]

Automatic Metrics. For automatic evaluation, we use several reference-based metrics , including BLEU-4 (Papineni et al., 2002), ROUGE-L (Lin, 2004), and BERTScore-F1 (Zhang et al., 2020a), computed using the `GEM-metrics` tool.[6] Following the recommendation in Zhang et al. (2020a), we use the re-scaled version of BERTScore which is easier to interpret. We additionally consider exact match scores, i.e., **R@K**, which measure whether one of the top K response candidates exactly matches the reference. Both **R@1** and **R@5** scores are reported. We lowercase all the strings and remove any extra spaces while computing the exact match between two strings.

Human Evaluation. It is well-known that popular automatic evaluation metrics may not always reflect the true quality of the generated responses (Celikyilmaz et al., 2021). Thus, we further carry out human evaluation on 297 examples randomly sampled from the validation data. Specifically, for each generated response, we collect human judgments on three questions: **grammaticality** (*"has the virtual assistant made any grammar errors?"*), **relevance** (*"has the virtual assistant misunderstood the user's request?"*), and **truthfulness** (*"has the virtual assistant provided any incorrect information as judged using the database and timestamp?"*). Three judgments are collected for each question, and we report the percentage of examples where *"no"* is the majority-voted answer. Higher percentages are better. Crowdworkers are recruited from Amazon Mechanical Turk with qualification requirements such as having a work approval rate higher than 80% and having performed a minimum of 100 annotations. They are paid at the rate of \$0.15 per judgment. For responses generated by the constrained decoding approach, the inter-annotator agreements for the three questions are around 90%, 78% and 76%, respectively, as measured by the percentage of examples where all three workers choose the same answer. More details are provided in Appendix A.

5.2 Main Results

Our main evaluation results on SMCalFlow are shown in Table 1. The first baseline we considered is to randomly sample responses from the generated QCFG. The other baseline is unconstrained LM decoding without using dataflow transduction. Model outputs are compared to human-authored agent utterances. For both unconstrained and constrained decoding, we prompt the LM with a string representation of the computation graph (i.e., the format released in SMCalFlow v2.0), followed by its execution result rendered as a JSON string. We use beam search with a beam size $K = 5$. The LM is initialized from CodeT5-base (Wang et al., 2021) and fine-tuned on all training examples. See Appendix B for more details.

As expected, the QCFG random sampling baseline struggles on all the automatic metrics, since dataflow transduction rules are written with an emphasis on truthfulness rather than fluency. This is reflected in the grammaticality score from the

[4] The "gold" responses are generated from a production system that includes rule-based constraints and manually validated by human experts, according to the dataset authors.

[5] Some of our rule bodies chose to expand the dataflow graph by calling functions, so we also had to implement those functions. In an end-to-end dialogue system, most of those functions would already have been implemented to support agent actions, not just natural language responses.

[6] https://github.com/GEM-benchmark/GEM-metrics

human evaluation as well. However, the truthfulness score is as high as 92.3%, indicating the generated responses are rarely incorrect. Generated responses are sometimes generic or omit relevant information for the user request, which partially contributes to the high truthfulness score, but is reflected in the relevance score, which is the lowest among all compared approaches.

In contrast, unconstrained decoding without dataflow transduction achieves impressive scores on automatic evaluation. Human evaluation also suggests that the generated responses are grammatically correct and relevant to the user's request in most cases. However, unconstrained decoding scores low on truthfulness, making false statements in about one-fifth of the generated responses. This high rate of factual errors from neural LMs is consistent with findings in prior work (Wiseman et al., 2017; Maynez et al., 2020). It is usually unacceptable in real-world applications.

Compared with unconstrained decoding, our proposed QCFG-constrained decoding achieves significantly better scores on exact match, truthfulness, and even relevance, while maintaining similar scores on BLEU, ROUGE, BERTScore and grammaticality. In particular, human evaluation results indicate that the quality of generated responses is very close to that of the gold responses. We share some qualitative analysis in Appendix C.

Since even the gold responses did not achieve 100% on human evaluation scores, we manually inspected those problematic examples. There are 4 examples for which the majority-voted answer to the ungrammaticality question is *"yes but understandable"*, and others are all rated as not containing any grammar errors. For the relevance question, 4 examples are due to arguably bad data and 2 examples receive tied votes. For the truthfulness question, 9 examples are due to arguably bad data, 8 examples are due to to crowd worker mistakes, and 6 examples receive tied votes.

5.3 Ablation Study

We next analyze how the amount of fine-tuning data and the context used in the input sequence impact the quality of generated responses. Results are summarized in Table 2.

Impact of fine-tuning: Without fine-tuning the LM, neither unconstrained nor constrained decoding works well. This is likely due to the mismatch between the pre-training tasks and the response generation task. However, after fine-tuning on only a random 3% of the training data, both approaches achieve significantly better scores, with larger gains on QCFG-constrained decoding. This suggests that QCFG-constrained decoding is much more data-efficient in the low-data regime. Indeed, using 3% of the training data, QCFG-constrained decoding is on par with the unconstrained decoding with 100% of the training data, indicating that several expert hours spent on creating dataflow transduction rules can dramatically reduce the cost of collecting training data. While gaps between unconstrained and QCFG-constrained decoding on automated metrics are small in the full-data setting (Table 1), unconstrained decoding still performs poorly on the truthfulness evaluation. Thus, truthfulness failures from unconstrained decoding are not straightforwardly solved by scaling up training data; QCFG-constrained decoding offers an easier path to faithful response generation.

	BLEU	**ROUGE**	**BERTSc.**	**R@1**	**R@5**
1. LM without fine-tuning					
✗	.45	.03	−.29	.00	.00
✓	.38	.22	.07	.02	.02
2. LM fine-tuned on 3% training data					
✗	.68	.82	.80	.26	.40
✓	.73	.83	.81	.39	.61
3. LM fine-tuned on full training data					
✗	.77	.86	.87	.47	.66
✓	.80	.87	.86	.56	.78
4. LM input without execution results					
✗	.58	.70	.72	.27	.42
✓	.78	.86	.84	.54	.77
5. LM input with user utterance					
✗	.76	.88	.87	.45	.65
✓	.77	.85	.85	.54	.78

Table 2: SMCalFlow ablation results, varying the amount of fine-tuning data (groups 1–3) and the context used in the input sequence (groups 4–5). ✗ and ✓ on the first column use unconstrained and QCFG-constrained decoding, respectively.

Impact of context: Results in groups 3–5 in Table 2 all use the full training data to fine-tune the LM. The difference is in the context used in the input sequence to the LM. For group 3, the input sequence is the computation concatenated with the execution result, which is the same setup used in §5.2. For group 4, we omit the execution results from the LM input (but not from the decoder constraints), whereas for group 5, we add the user utterance (prefixed to the computation). Comparing group 3 and group 4, omitting execution results significantly harms the performance of unconstrained decoding. In contrast, dataflow transduction rules can execute the computation internally, and do not require the LM to condition on it. Comparing group 3 and group 5, adding user utterances to prompts does not bring any additional benefits to both approaches.

5.4 Experiments with MultiWOZ Dataset

To demonstrate the general applicability of our approach for response generation, we carry out a brief study on the widely used MultiWOZ 2.1 dataset (Budzianowski et al., 2018; Eric et al., 2020). We automatically convert the system act annotations to dataflow computations and write 14 transduction rules. For generating responses, we use the predicted system acts from the MT-TOD system (Lee, 2021). Similar to our experiments on SMCalFlow, we fine-tune CodeT5-base on all training examples, using the ground-truth belief state and predicted system act as the input sequence. For evaluation, we randomly sample 100 examples from the test split, and two authors manually rate the generated responses from our QCFG-constrained decoding system and the MT-TOD system. The agreement is 100%. Almost all generated responses are grammatically correct and relevant to the user utterance. To rate truthfulness, we use the predicted system acts as the references. Our QCFG-constrained decoding approach produce truthful responses for all 100 examples, whereas only 89 responses from the MTTOD system are truthful with respect to its predicted system act. Among the 11 remaining examples, 7 of them are due to imperfect delexicalization and 4 are due to hallucination.

6 Related Work

One line of response generation research focuses on generating fluent and coherent responses directly from user utterances without any intermediate structured representation. This paradigm is mostly used for chatbots, as in early rule-based systems (Weizenbaum, 1966; Wallace, 2009), neural conversation models (Vinyals and Le, 2015; Shang et al., 2015; Sordoni et al., 2015; Li et al., 2016; Serban et al., 2016), and recent large-scale pretrained LMs like DialoGPT (Zhang et al., 2020b) and GPT-3 (Brown et al., 2020).

Another line focuses on generating text from structured data, with applications beyond dialogue response generation. For example, the WebNLG challenge (Gardent et al., 2017) generates natural language descriptions from relation tuples, and Lebret et al. (2016) generate a biography from a structured "infobox" record. Many recent dialogue response generation tasks adopt dialogue-act-based meaning representations, including the MultiWOZ dataset (Budzianowski et al., 2018), the Schema-Guided dialogue dataset (Rastogi et al., 2020), and the E2E NLG challenge (Dusek et al., 2020). In contrast, our response generation task uses computations as the input, which do not directly encode the dialogue acts of the responses. This is a more challenging task, as the system needs to perform extra reasoning to obtain the derived information. In this sense, our task is similar to the one in CoSQL (Yu et al., 2019) and Logic2Text (Chen et al., 2020).

Constrained decoding techniques for neural LMs have been developed for text generation with different types of constraints (Balakrishnan et al., 2019; Dathathri et al., 2020; Lu et al., 2021, 2022). Shin et al. (2021) develop a constrained decoding approach for semantic parsing by restricting the LM output at each step according to a given grammar. Differently, the grammar productions in our case are derived dynamically for each input.

7 Conclusion

We have described a hybrid approach for building dialogue response generation systems. Our approach introduces a new formalism for transducing a dataflow graph into a QCFG, which is then used in a constrained decoder that intersects the QCFG with a neural LM. (In future work, the QCFG could be weighted to express its own preferences.) This formal framework makes it possible to write rules to precisely and truthfully describe data and its provenance while deferring surface realization decisions to a flexible language model.

This new approach outperforms unconstrained conditional language modeling in both automatic and human evaluations, especially on truthfulness. Moreover, using 3% of the training data, the constrained decoding approach is on par with the unconstrained decoding approach when it uses 100% of the training data, indicating that several expert hours spent on authoring rules can dramatically reduce the cost of data annotation.

References

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 831–844. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2021. Evaluation of text generation: A survey. *arXiv:2006.14799v2 [cs.CL]*.

Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. 2020. Logic2Text: High-fidelity natural language generation from logical forms. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2096–2111, Online. Association for Computational Linguistics.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *Proceedings of 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Ondrej Dusek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG challenge. *Computer Speech and Language*, 59:123–156.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133. Association for Computational Linguistics.

Paul Grice. 1975. Logic and conversation. In *Syntax and semantics*, volume 3, pages 41–58. Academic Press.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 704–710, Montreal, Quebec, Canada. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1203–1213. The Association for Computational Linguistics.

Yohan Lee. 2021. Improving end-to-end task-oriented dialog system with a simple auxiliary task. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1296–1303, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of International Conference on Learning Representations*.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. 2022. NeuroLogic A*esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 780–799, Seattle, United States. Association for Computational Linguistics.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. NeuroLogic decoding: (Un)supervised neural text generation with predicate logic constraints. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4299, Online. Association for Computational Linguistics.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Emmanouil Antonios Platanios, Adam Pauls, Subhro Roy, Yuchen Zhang, Alex Kyte, Alan Guo, Sam Thomson, Jayant Krishnamurthy, Jason Wolfe, Jacob Andreas, and Dan Klein. 2021. Value-agnostic conversational semantic parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8689–8696.

Ehud Reiter. 2022. What are the problems with rule-based NLG? `https://ehudreiter.com/2022/01/26/problems-with-rule-based-nlg/`.

Subhro Roy, Sam Thomson, Tongfei Chen, Richard Shin, Adam Pauls, Jason Eisner, and Benjamin Van Durme. 2022. BenchCLAMP: A benchmark for evaluating language models on semantic parsing. *arXiv:2206.10668 [cs.CL]*.

Semantic Machines, Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the AAAI Conference on Aritificial Intelligence*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586, Beijing, China. Association for Computational Linguistics.

Richard Shin, Christopher H. Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7699–7715. Association for Computational Linguistics.

David Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 23–30, New York City. Association for Computational Linguistics.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, Colorado. Association for Computational Linguistics.

Oriol Vinyals and Quoc Le. 2015. A neural conversation model. In *Proceedings of ICML Deep Learning Workshop*.

Marilyn A. Walker, Owen C. Rambow, and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech & Language*, 16(3):409–433. Spoken Language Generation.

Richard S. Wallace. 2009. The anatomy of A.L.I.C.E. In *Parsing the Turing Test*, pages 181–210. Springer, Dordrecht.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Joseph Weizenbaum. 1966. ELIZA – a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. BERTScore: Evaluating text generation with BERT. In *Proceedings of the 8th International Conference on Learning Representations, ICLR 2020*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020b. DIALOGPT: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

A Human Evaluation Details

A screenshot of the MTurk interface for human evaluation is shown in Fig. 4. The inter-annotator agreements for different systems are provided in Table 3. It can be observed that the gold responses receive the highest agreements on all three questions. The QCFG-constrained decoding has slightly higher agreements than the unconstrained dcoding. The QCFG random sampling receives a significantly lower agreement on "Grammaitical", which is likely because this approach may produce ungrammatical responses but people may not agree on whether these are understandable.

B Model Configurations

For SMCalFlow, we fine-tune the CodeT5 model for a fixed number of epochs (=10). For MultiWOZ, we fine-tune the model for at most 10 epochs and do early stopping based on the on the loss on the development set.We use the AdamW optimizer (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, using a linear learning rate scheduler with an initial learning rate of 5×10^{-5}. For decoding, we always use a fixed beam size of 5.

The CodeT5-base models used in our experiments have 220 million parameters. We used machines with 32GB V100 GPUs for model fine-tuning while the decoding experiments were carried out on CPU-only machines.

For SMCalFlow experiments, the input sequence to the LM is the string representation of the computation in the lispress format followed by its the execution result is rendered as a JSON string, e.g., "*Plan: (Yield (Event.start (...))) Result: {"type": "DateTime", "value": ... } <s>*", where the last token is a special token to separate the input and the output. For the ablative study (group 5) in §5.3, the user utterance is prefixed to the sequence, e.g., "*User: When do I have thee oil change on my car scheduled for? Plan: ... Result: ... <s>*".

For MultiWOZ experiments, the computation is rendered as a raw JSON string which encodes the ground-truth belief state and the predicted system act. There is no execution result for these computations.

C Qualitative Analysis

We looked at 100 randomly selected examples from the experiments on SMCalFlow from §5.2, and compare the generated responses from both unconstrained decoding and QCFG-constrained decoding with the human-annotated gold responses provided by the dataset. We summarize the differences between the generated and gold responses in Table 4, using the following categories:

Untruth The system reports incorrect information.

Omission The system fails to mention information mentioned in the gold response.

Addition The system mentions additional (correct) information that is not mentioned in the gold response.

Minor Difference The system uses a different phrasing than the gold response that nonetheless has the same information and fluency.

Disfluency The system output is disfluent.

Annotation Error The system output is acceptable but the gold annotation contains a fluency or factuality error.

For unconstrained decoding, 57 out of 100 responses differ from the gold responses, whereas for QCFG-constrained decoding, only 51 of 100 responses differ. This result is consistent with the R@1 column of Table 1 (mismatch rates of 53% and 44% respectively on the full validation set).

As expected, the most noticeable difference is the number of Untruths reported by the unconstrained system – 19%, close to the 18% rate found in the human evaluations in Table 1. We show some examples of Untruths in Fig. 5. The QCFG-consrained system produed no Untruths. Conversely, the QCFG-constrained system produces substantially more Omissions than the unconstrained system. Of the 11 omissions produced by the constrained system, 3 are are identical to the unconstrained output while 7 are on inputs for which the unconstrained output produce an Untruth. In other words, our system successfully removed the 19 Untruths by the system, but in 7 of those cases, it produced a shorter (but still factually correct) input than the preferred gold annotation for that example. We also note that the gold dataset is not consistent in how much information is included in the responses – short answers like *"Looks like it"* in Example C from Fig. 5 are present in the gold annotations on examples similar to Example C. Furthermore, both

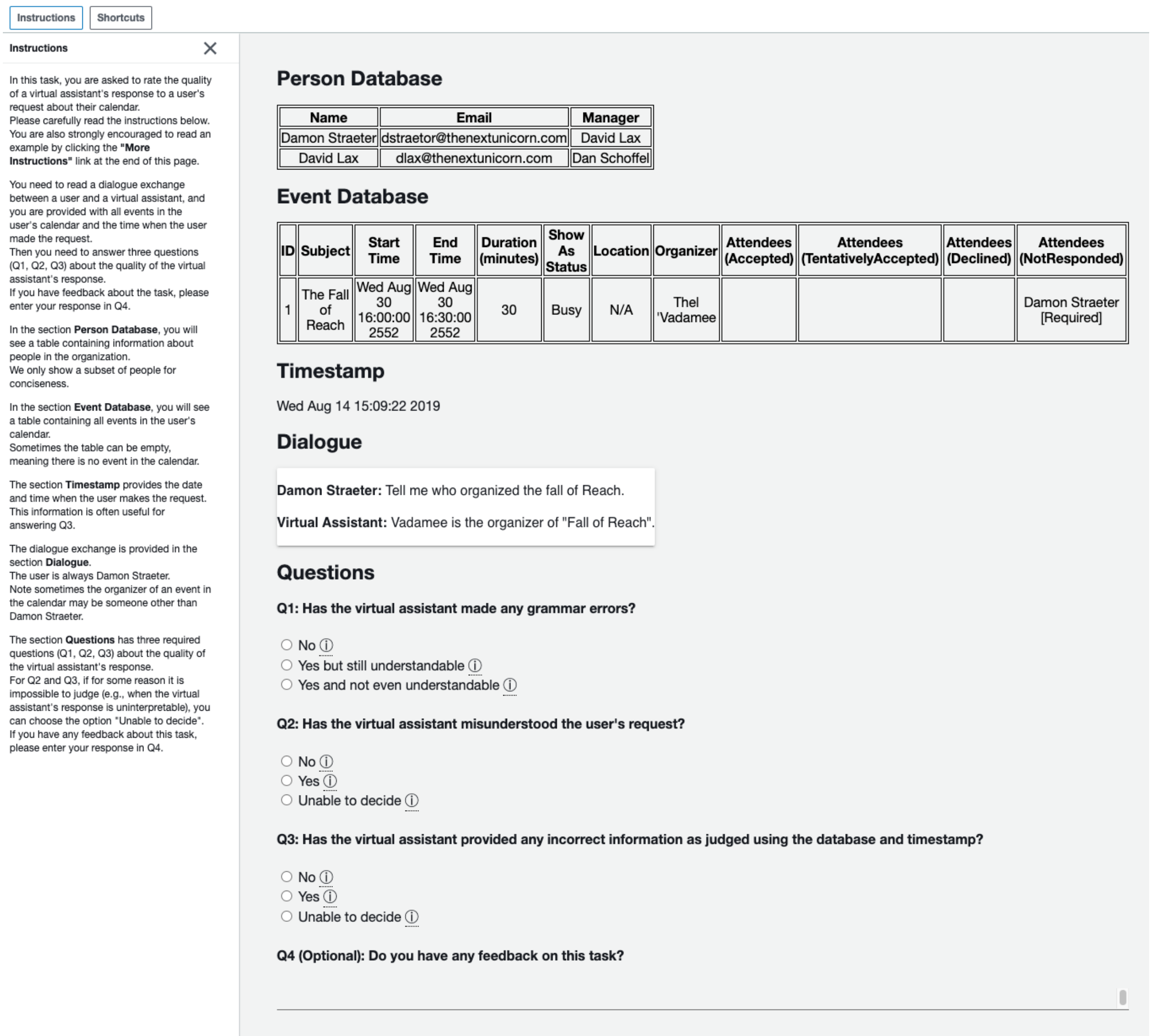

Figure 4: A screenshot of the MTurk interface for human evaluation.

systems produce more Additions than Omissions, indicating that there is not a systematic bias towards shorter answers overall. In future work, the model could be made to select more descriptive responses by adding a brevity penalty in the decoder or by weighting the QCFG productions, so that responses are scored not only by the LM but also by the QCFG.

D Limitations and Future Direction

Authoring transduction rules is relatively easy but may be still labor intensive for complex domains. Future work might explore (semi-)automatically deriving transduction rules from data or learning to synthesize them from domain specifications, or curating a collection of domain-general transduction rules which can be directly used in new domains.

Another direction would be to extend the dataflow transduction rules so they can encode pragmatic knowledge and context-dependent policies. For example, a dataflow transduction rule could call a neural network to assess the suitability of applying the rule to a given node in the dataflow graph, and weight the resulting QCFG production accordingly.

E Dataset License

The SMCalFlow dataset is distributed under the CC BY-SA 4.0 license. To the best of the authors knowledge, the MultiWOZ datasets were released under MIT license as shown in https://github.com/budzianowski/multiwoz. Our experiments follow the intended use of these datasets, which is to advance research in dialogue systems.

System	Grammatical	Relevant	Truthful
QCFG Random Sampling	.58	.75	.71
Unconstrained Decoding	.86	.71	.71
QCFG-Constrained Decoding	.90	.78	.76
Gold	.95	.81	.80

Table 3: The inter-annotator agreements for different systems.

	Unconstrained	Constrained
Untruth	19	0
Omission	3	11
Addition	17	18
Minor Difference	10	13
Disfluency	1	1
Annotation Error	7	8
Total	57	51

Table 4: Classification of differences between generated responses and human-annotated gold responses on 100 randomly sampled examples from the SMCalFlow dataset. Details are provided in Appendix C.

	User Utterance	Reference Agent Utterance	QCFG-Constrained decoding	Unconstrained decoding
A.	Do I have any events scheduled on Sunday the 3rd?	No, I didn't find any events on March 3rd.	No, I didn't find any events on March 3rd.	No, I didn't find any events on **May 3rd**.
B.	Does Pikachu fly in tomorrow?	Yes, I found one event matching "fly in" with Pikachu tomorrow. It's "fly in" from 4:00 to 4:30 PM.	Yes, I found one event matching "fly in" with Pikachu tomorrow. It's "fly in" from 4:00 to 4:30 PM.	Yes, I found one event matching "fly in" with Pikachu tomorrow. It's "Fly In" from **5:00 to 5:30 PM.**
C.	Can you find me any events during last March?	Yes, I found one event last March. It was "Dancing in Athenry" on March 17, 2019 from 4:00 to 4:30 PM.	**Looks like it.**	Yes, I found one event last March. It was "**Erin Go Bragh**" on March 17, 2019 from 5:00 to 5:30 PM.

Figure 5: Example predictions from fine-tuned CodeT5 model with QCFG-constrained decoding and with unconstrained decoding. In all the examples shown, outputs from unconstrained decoding are untruthful to the database due to content hallucination even though the model has access to the correct execution results as part of the input. We observe that in a few cases, the constrained model prefers truthful but pragmatically unhelpful omissions like such as *"Looks like it"* (in Example C) compared to a more specific response.