Towards Transparent Interactive Semantic Parsing via Step-by-Step Correction

Anonymous ACL submission

Abstract

Existing studies on semantic parsing focus on mapping a natural-language utterance to a logical form (LF) in one turn. However, because natural language may contain ambiguity and variability, this is a difficult challenge. In this work, we investigate an interactive semantic parsing framework that explains the predicted LF *step by step* in natural language and enables the user to make corrections through *natural-language feedback* for individual steps. We focus on question answering over knowledge bases (KBQA) as an instantiation of our framework, aiming to increase the transparency of the parsing process and help the user trust the final answer. We construct INSPIRED, a crowdsourced dialogue dataset derived from the COMPLEXWEBQUESTIONS dataset. Our experiments show that this framework has the potential to greatly improve overall parse accuracy. Furthermore, we develop a pipeline for dialogue simulation to evaluate our framework w.r.t. a variety of state-of-the-art KBQA models without further crowdsourcing effort. The results demonstrate that our framework promises to be effective across such models.[1]

1 Introduction

Semantic parsing aims to map natural language to formal meaning representations, such as λ-DCS, API calls, SQL and SPARQL queries. As seen in previous work (Liang et al., 2013; Yih et al., 2014, 2015; Talmor and Berant, 2018b; Chen et al., 2019; Lan and Jiang, 2020a; Gu et al., 2021), parsers still face major challenges: (1) the accuracy of state-of-the-art parsers is not high enough for real use, given that natural language questions can be ambiguous or highly variable with many possible paraphrases, and (2) it is hard for users to understand the parsing process and validate the results.

In response to the challenges above, recent work (Li and Jagadish, 2014; He et al., 2016; Chaurasia and Mooney, 2017; Su et al., 2018; Gur et al., 2018; Yao et al., 2019a; Elgohary et al., 2020) explores *interactive semantic parsing*, involving human users to give feedback and boost system accuracy. For example, Su et al. (2018) show that fine-grained user interaction greatly improves the usability of natural language interfaces to Web APIs. Yao et al. (2019a) allow their semantic parser to ask users clarification questions when generating an If-Then program. And recently, Elgohary et al. (2020) crowdsources the SPLASH dataset for correcting SQL queries using natural language feedback.

Compared with these approaches, we aim to enhance the transparency of the parsing process and increase user confidence in the final answer. Figure 1 shows a desired dialogue between user and agent. We design an interactive framework for semantic parse correction that can explain the predicted complex logical form (LF) in a step-by-step manner and enable the user to make corrections to individual steps in natural language. To demonstrate the advantages of our interactive framework, we propose

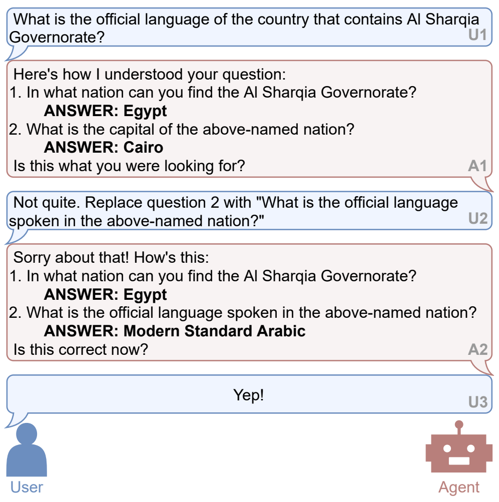

What is the official language of the country that contains Al Sharqia Governorate?
U1
Here's how I understood your question:
1. In what nation can you find the Al Sharqia Governorate?
ANSWER: Egypt
2. What is the capital of the above-named nation?
ANSWER: Cairo
Is this what you were looking for?
A1
Not quite. Replace question 2 with "What is the official language spoken in the above-named nation?"
U2
Sorry about that! How's this:
1. In what nation can you find the Al Sharqia Governorate?
ANSWER: Egypt
2. What is the official language spoken in the above-named nation?
ANSWER: Modern Standard Arabic
Is this correct now?
A2
Yep!
U3
User
Agent

Figure 1: Example dialogue from our dataset (dubbed INSPIRED). The agent turns (A_i's) illustrate our emphasis on transparency by explaining the predicted logical form step by step in natural language, along with intermediate answers, to the user for feedback.

[1]Our code and dataset will be released upon acceptance.

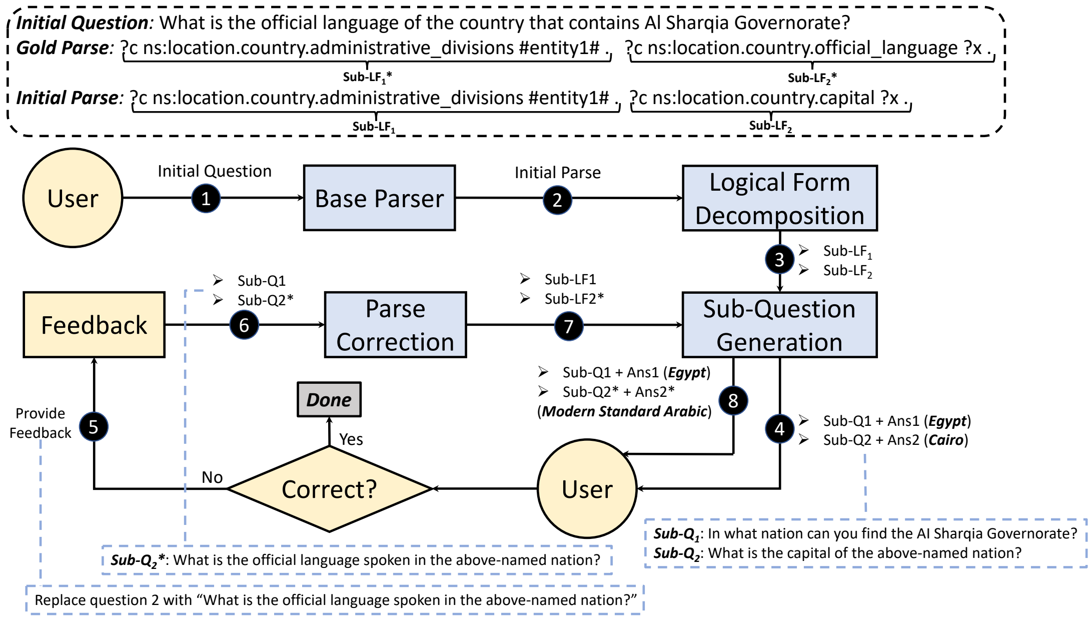

Initial Question: What is the official language of the country that contains Al Sharqia Governorate?
Gold Parse: ?c ns:location.country.administrative_divisions #entity1# . ?c ns:location.country.official_language ?x .
Sub-LF1*
Sub-LF2*
Initial Parse: ?c ns:location.country.administrative_divisions #entity1# . ?c ns:location.country.capital ?x .
Sub-LF1
Sub-LF2
User
Initial Question
Base Parser
Initial Parse
Logical Form Decomposition
Sub-LF1
Sub-LF2
Feedback
Sub-Q1
Sub-Q2*
Parse Correction
Sub-LF1
Sub-LF2*
Sub-Question Generation
Sub-Q1 + Ans1 (Egypt)
Sub-Q2* + Ans2* (Modern Standard Arabic)
Done
Yes
Provide Feedback
No
Correct?
User
Sub-Q1 + Ans1 (Egypt)
Sub-Q2 + Ans2 (Cairo)
Sub-Q1: In what nation can you find the Al Sharqia Governorate?
Sub-Q2: What is the capital of the above-named nation?
Sub-Q2*: What is the official language spoken in the above-named nation?
Replace question 2 with "What is the official language spoken in the above-named nation?"

Figure 2: Illustration of our interactive semantic parsing framework for KBQA. The box on the top lists a running example. The prefix of a SPARQL query (i.e., LF used for KBQA in this paper) in this example is omitted for brevity. The figure on the bottom shows the entire workflow of our framework using the example above.

to instantiate it for complex question answering over knowledge bases (KBQA), where interactive semantic parsing has remained largely unexplored.

Figure 2 illustrates our framework with a concrete example: A base parser predicts an initial parse, which we decompose into sub-LFs and translate to natural-language questions (i.e., *Sub-Question Generation*). This shows the steps of answering the question, allowing the user to see exactly how a final answer is found and be confident that it is correct or give feedback in natural language to correct the steps. If any user feedback is given, our framework uses it to correct errors in the current parse (i.e., *Parse Correction*).

To build models for Sub-Question Generation and Parse Correction, we construct a dataset via crowdsourcing, based on the COMPLEXWEBQUESTIONS (CWQ) dataset (Talmor and Berant, 2018b), which is widely used for complex QA. To make LFs understandable to crowdworkers, we translate each sub-LF into a templated sub-question using a rule-based method. During crowdsourcing, workers paraphrase the templated question into a natural one. We create a dialogue for each complex question, an example of which is shown in Figure 1. Our dataset, dubbed INSPIRED (**IN**teractive **S**emantic **P**ars**I**ng for Cor**RE**ction with **D**ecomposition), will facilitate further exploration of interactive semantic parsing for KBQA.

Our main contributions are as follows: (1) We design a more transparent interactive semantic parsing framework that explains to a user how a complex question is answered step by step and enables them to make corrections in natural language and trust the final answer. (2) To support research on interactive semantic parsing for KBQA, we release a high-quality dialogue dataset using our framework. (3) We establish baseline models for two core subtasks in this framework: Sub-Question Generation and Parse Correction. (4) Although INSPIRED is constructed using a selected base parser, we are able to train models to simulate user feedback, allowing us to study the promise of our framework to correct errors made by other semantic parsers without more annotation effort. With these contributions, we hope to inspire many directions of future work, which we discuss in the end.

2 Dataset Construction

In this section, we describe the workflow for dataset construction following the design of our framework (Figure 2). We prepare pairs of complex questions and SPARQL parses predicted by a base semantic parser (Section 2.1.1). Then, we decompose the gold and predicted parses and determine *correction operations* (Section 2.1.2). The sub-LFs are translated to questions using templates (Section 2.1.3) and we employ crowdworkers to paraphrase these questions using natural language (Section 2.2).

2.1 Dialogue Preparation for Crowdsourcing

2.1.1 Preparing Questions and SPARQL

We start with the ComplexWebQuestions 1.1 (CWQ) dataset (Talmor and Berant, 2018a,b), which contains complex questions paired with gold SPARQL queries for Freebase (Bollacker et al., 2008). We adopt a transformer-based seq2seq model (Vaswani et al., 2017) as the base semantic parser to prepare a predicted SPARQL query for each complex question (see the second and third paragraphs in Section 4 for our rationale).

As a simplifying assumption, we take gold named entities mentioned in a question as given. Specifically, we replace named entities in a SPARQL query with special tokens such as *#entityX#*, where X is a number corresponding to the order in which the entity appears. After parsing, we replace these tokens with the gold entities. The challenge of addressing errors caused by named entity recognition and linking in a real KBQA system is left as an important piece of future work.

In order to reduce data collection cost, we select a subset of questions in the training data of CWQ to create dialogues in INSPIRED's training set. We conduct an analysis of repeated predicates and question types, and ensure that each predicate occurs at least three times in INSPIRED's training set when possible. We include every question where the base parser makes an error and ensure coverage of the four multi-hop reasoning types (Talmor and Berant, 2018b). Different reasoning types require different translation strategies in order to represent their logical forms in English (see Section 2.1.3 and Appendix A.3). We create 10,374 dialogues in total, based on 3,492 questions from the training set, 3,441 from the validation set, and 3,441 from the test set of CWQ. We omit a small set of questions from the original validation and test sets that are consistently confusing to crowdworkers. Table 1 shows a breakdown of the CWQ question types in the INSPIRED dataset, along with the average number of corrections and sub-questions.

2.1.2 Logical Form Decomposition

An important goal of creating INSPIRED is to make the process of question answering transparent to the user. Each dialogue features a decomposition process by which our framework transforms the complex question into an initial parse, breaks it into sub-LFs, retrieves answers, and presents this whole process to the user for correction. The overarching strategy of the decomposition process is to identify the predicates that express distinct components in the LF of the complex question, which correspond to individual sub-questions. Typically, these components appear as a *triple* in the logical form such as *Sub-LF*$_1$ in Figure 2, which is comprised of a head entity, a predicate, and a tail entity. Logical forms in CWQ typically contain two or three of these components. There can be multiple predicates that group together to express one component, for example those connected by a CVT (Compound Value Type) node, in which case the two predicates and their two entities will form one component. Within these, there can be filters and/or restrictions, which provide additional information about entities of the main predicate and are typically merged to the corresponding component.

Number of	Train	Dev	Test	Overall
Complex Questions	3,492	3,441	3,441	10,374
- Composition	1,196	1,532	1,490	4,218
- Conjunction	1,796	1,503	1,553	4,852
- Comparative	253	217	207	677
- Superlative	247	189	191	627
Predicted Sub-Questions	1.7	2.0	1.9	1.9
Gold Sub-Questions	2.2	2.1	2.1	2.1
Range of the number of predicted sub-questions				0 - 5
Range of the number of gold sub-questions				2 - 4
Average number of edits				1.4
Dialogues with 0 edits				5,016

Table 1: Statistics for our INSPIRED dataset: the number of complex questions for each reasoning type, the average number of sub-questions and edit operations in a dialogue (excluding those that do not have edits).

Using this strategy, we decompose both the parser's predicted SPARQL query and the gold one into sub-LFs, and compare those sub-LFs to determine the sequence of operations needed to transform the predicted parse into the gold parse, including inserting, deleting, or replacing a sub-LF, which is to be paraphrased by crowdworkers (Section 2.2) based on our templated sub-question. These operations determine the "correction" steps in each dialogue, where the agent asks the user if any corrections are needed (Figure 1, turn A1), and the user either confirms that the initial parse is correct or provides corrections (turn U2). Though any new sub-questions that are introduced use natural and varied language, the correction operations are given using templates (i.e., *replace question #X with Y, delete question #X, insert question Y*). More details about how dialogues are formed around complex questions can be found in Appendix A.1.

2.1.3 Explaining SPARQL

After the LF decomposition, we develop a strategy for how to represent the sub SPARQL queries in a form that crowdworkers can understand. Thus we create a template corpus and a rule-based translation method to do so. The corpus consists of 772 different predicates that appear in the CWQ dataset and translations of each into a basic template that conveys the content. More details about the translation of LFs with different reasoning types into sub-questions are found in Appendix A.2 and A.3.

2.2 Crowdsourcing

To make queries understandable for an average user, as in Figure 1, we translate the decomposed LFs into English questions using templates as mentioned in Section 2.1.3. To obtain natural sounding questions, we conduct crowdsourcing on Amazon Mechanical Turk (AMT), in which crowdworkers are employed to rephrase sub-questions from the clunky, templated form into more concise and natural English in the context of a dialogue. The task is conducted using ParlAI (Miller et al., 2017), which allows us to set up a versatile dialogue interface.

In each dialogue, every turn of the interlocutors has prescribed content. A total of 14 crowdworkers are employed to express the content in natural language and complete a maximum of 1,800 dialogues. Because the crowdsourcing task for this dataset requires extensive, detailed instructions, we design the task quite carefully with multiple stages of checkpoints to ensure quality of data collection. An overview of these phases can be seen in Table 2 and other details are presented in Appendix A.4. We recruit and retain a small set of exemplary workers for this task (see item 4 in General Principles in Table 2). This phased strategy, while requiring more effort, proves to be effective in ensuring overall data quality which will be shown in Section 3.

Phased Crowdsourcing Protocol

Phase 1: Tutorial
1. Worker reads examples and explanations of the task.
2. Worker receives specific instructions for how to rephrase questions of different types.

Phase II: Qualification Quiz
1. Worker completes an 8-question multiple choice quiz. Quiz questions are based on the tutorial content.
2. Worker must achieve at least 7 out of 8 to pass. They may take the quiz more than once, but there is a ten minute wait period between attempts.

Phase III: Trial Period
1. Worker completes 10 predetermined tasks which were chosen as representative examples for all the tasks.
2. Tasks are manually graded. If the work is overall good, the worker receives specific feedback on anything that was done incorrectly.
3. If quality is not good, worker is eliminated.
4. Workers get paid the regular rate for each task and upon completing the 10 tasks, receive a bonus for the time spent on the tutorial and qualification quiz.

Phase IV: Batches of Tasks
1. Worker is given access to a batch of 100 tasks, which are spot-checked for quality. A bonus is given as the worker passes each set of 100 tasks.
2. If quality is good, workers are given a second batch of 100 questions, also spot checked.
3. Batch size increases based on worker quality and speed.
4. Worker completes up to 1800 tasks.

General Principles
1. Prompt feedback, payment, and release of new batches
2. Provide a link to the tutorial so that it can be accessed at any time.
3. Higher than average payment.
4. Keep pool of workers small for better communication and quality control.
5. Verify that workers are native English speakers.

Table 2: The phased crowdsourcing protocol for our Amazon Mechanical Turk task.

	Template Corpus	**Rephrased Corpus**
Avg Length	17.3	10.7
Unigrams	8,465	9,864
Bigrams	21,072	44,085
Trigrams	31,838	81,479

Table 3: Comparison of average length (in words) of templated and rephrased questions as well as the size of vocabulary for 1-, 2-, and 3-grams across all templates and rephrased questions, demonstrating the increased diversity of rephrased questions.

3 Dataset Analysis

In this section, we conduct a thorough quality analysis of INSPIRED dataset and highlight aspects that contribute to overall quality, including paraphrasing characteristics and contextual awareness.

Overall Data Quality. In each dialogue, the crowdworker is required to rephrase the original complex question and each templated sub-question. Overall, we believe the quality of the data to be high for a few reasons. In the collection process, our crowdworkers read a detailed tutorial, pass two qualification tasks, and have their work spot-checked at each stage of collection. Because we keep our pool of workers small, we are able to maintain frequent communication with them throughout the process, giving feedback in an ongoing fashion.

Furthermore, a semi-automatic data cleaning method is used to identify inaccurate paraphrases, which are manually repaired. This results in edits to 325 sub-questions. Based on observation on a held-out subset of the data, we estimate that only 3.1% of sub-questions still have inaccuracies, after cleaning. More details are in Appendix B.1.

Paraphrasing Characteristics. Table 3 shows the difference between the vocabularies (unique words) of all the templates in INSPIRED and the rephrased versions of sub-questions, which are calculated using GEM evaluation scripts (Gehrmann et al., 2021). Further, the mean length of the templated questions is 17.3 words, while the mean length of the rephrased questions is 10.7 words. These comparisons demonstrate that the rephrased

	ROUGE-1	ROUGE-2
Random Context	22.8	3.4
Actual Context	**27.7**	**6.2**

Table 4: Comparison of the n-gram overlap between the paraphrase and the context for a sub-LF vs. other randomly chosen context for the same sub-LF.

questions show much more diversity in phrasings and lexical choices, but are also more concise. More GEM metrics can be seen in Appendix B.2.

In order to better understand how crowdworkers rephrased templates, 100 randomly selected sub-questions are studied in terms of lexical relationships between the template and rephrased versions. We find that they are using synonmy, hypernymy and hyponymy in rephrasings of the templates, in addition to changing word order. This analysis can be found in Appendix B.3.

Contextual Awareness. Additionally, crowdworkers are encouraged to incorporate contextual information of a given sub-question into their rephrasings, thus improving the contextual richness of the dataset. In order to demonstrate contextual awareness, Table 4 shows the average ROUGE-1 and ROUGE-2 scores of all sub-questions in their actual contexts (the complex question and any preceding sub-questions), in comparison to the same sub-questions in a randomly assigned context that utilizes the same sub-logical form. Entities are masked with *#entity#* tokens to prevent the actual context from being advantaged by overlap in entity names. The higher scores for the actual context indicate that the wording of sub-questions reflect the context from which they are derived. Appendix B.4 provides more details on this aspect.

4 Experiments

In this section, we explore several base semantic parsers and show how we choose one as the initial parser to construct INSPIRED. Then, we conduct extensive experiments on those two core sub-tasks (i.e., sub-question generation and parse correction) in our framework. Finally, in order to study the promise of our framework for other parsers (beyond the one used to construct INSPIRED) without introducing extra crowdsourcing effort, we simulate dialogues based on our trained models for sub-question generation and parse correction. We train all models on 4 GTX 1080 Ti 11 GB GPUs.

Firstly, we explore Transformer (Vaswani et al., 2017), BART-large (Lewis et al., 2020) and QGG (Lan and Jiang, 2020b) as base parsers. In the official leaderboard[2] of CWQ, QGG is the best-performing method in the line of query graph generation approaches. Models like NSM+h (He et al., 2021) and PullNet (Sun et al., 2019) directly output final answers without LFs, which cannot be made more transparent or interactive with our framework. CBR-KBQA (Das et al., 2021) is the SOTA model on this dataset as of the submission time, but as its code is not available, we choose Transformer and BART-large as the two candidate parsers. We input the complex question to these two seq2seq models and output the LF. Since entities are masked in the LFs for these models, we provide QGG with gold entities for fair comparison. We report their LF exact match (EM) and F1 scores in Table 5.

Models	EM	F1
*Transformer (Vaswani et al., 2017)	52.3	58.6
BART-large (Lewis et al., 2020)	60.9	65.8
QGG (Lan and Jiang, 2020b)	-	49.0

Table 5: Performance of different semantic parsers on CWQ test set.[3] The asterisk (*) denotes the initial semantic parser we choose for constructing INSPIRED.

We finally select Transformer as the initial parser because it is neither state-of-the-art nor has overly poor performance. As the intention is to create a dataset that represents a wide range of parsing errors and correction strategies, a "middle-of-the-road" parser is best for achieving good coverage but also being of decent quality. We report the characteristics of errors made by Transformer in Appendix B.5. We will explore the other two models in Table 5 through simulation (Section 4.3).

In the following two sections, we explore two sub-tasks under our framework. We treat both of them as seq2seq tasks, then present and evaluate several baseline models including Seq2Seq (Sutskever et al., 2014), Transformers (Vaswani et al., 2017), BART-base and BART-large (Lewis et al., 2020) for each task, in which we use INSPIRED for training and testing. After that, we conduct error analysis for both sub-tasks by examining 100 examples respectively. Details of the analysis can be found in Appendix C.

[2]https://www.tau-nlp.org/compwebq-leaderboard

[3]Since INSPIRED excluded a small set of questions from CWQ, for fair comparison, scores here are calculated using questions in CWQ test set which are included in INSPIRED.

Correction Models	Turn-level EM	Dialog-level EM
w/o Correction	-	52.3
2nd-Beam	-	55.8
Seq2Seq(LSTM)	78.9	65.0
Transformer	81.2	68.0
BART-base	82.3	70.3
BART-large	**82.9**	**71.3**

Table 6: Turn-level and Dialogue-level accuracy of different models after incorporating feedback.

Context	Dialog-level EM	Turn-1 (3441)	Turn-2 (3441)	Turn-3 (345)	Turn-4 (56)
w/o Correction	52.3	-	-	-	-
BART-large					
w/o Context	71.3	84.6	81.5	85.5	53.6
+ h_q	72.2	84.7	82.2	89.3	**100.0**
+ h_{lf}	72.0	84.3	82.1	89.3	**100.0**
+ h_q & h_{lf}	**73.5**	**86.4**	**83.2**	**91.0**	**100.0**

Table 7: Parse correction performance when considering different contexts. h_{lf} and h_q denote the dialogue history of sub-LFs and sub-questions respectively.

4.1 Parse Correction with NL Feedback

Given a sub-question q, parse correction task is to convert it into a new sub-LF p. By parsing the templates used by correction operations as mentioned in Section 2.1.2, we extract the operation (i.e., replace, delete, or insert a sub-question) and apply it to the appropriate step. Then, sub-LFs will be compiled accordingly to form a correction parse P for the entire question. We predict the sub-LF based on q without considering contexts, and present the results of several baselines in Table 6. We report both the turn-level accuracy—the accuracy of sub-LFs in correction turns—and the dialog-level accuracy—the end-to-end accuracy of the entire LFs after correction—on our test set.

Since models like BART adopt a subword tokenization scheme, the validness of predicates generated by concatenating subwords can not always be guaranteed. We use beam search of size 10 to generate LFs as candidates, filtering those with invalid predicates and excluding erroneous predictions previously made by the parser. We additionally compare with a 2nd-Beam baseline, which applies beam search on the base parser to obtain two initial parses and uses the second for parse correction. It has some performance gains over the setting without correction, but is much lower than those settings with human feedback. Results in Table 6 further suggest: (1) incorporating human feedback can substantially improve the parse accuracy and (2) using BART-large with pretraining as the correction model achieves the best performance, achieving 19.0 points higher than the initial parser in terms of the dialog-level EM score.

Then, using BART-large as the correction model, we further study the correction process by concatenating different contexts to the input, including the history of sub-questions h_q and sub-LFs h_{lf}. We report both the accuracy for each turn of correction and the end-to-end accuracy. As shown in Table 7, we find that: (1) Adding contexts into the input can further improve the correction accuracy. (2) As the number of turns goes up, context contributes more to the correction process, which indicates that including the full dialogue history in the input leads to the best results. (3) The BART-large model with inputs that leverage h_q and h_{lf} achieves the best performance, with a 21.2 increase under dialog-level EM compared to the initial parser.

4.2 Sub-Question Generation

Sub-question generation aims to translate a sub-LF p into a natural sub-question q. Table 8 lists generation performance from five baselines without considering contexts. We explore an off-the-shelf paraphrasing model, which takes q^t as the input and outputs q. It is fine-tuned on BART-large using three paraphrasing datasets including Quora,[4] PAWS (Zhang et al., 2019) and MSR paraphrase corpus (Dolan and Brockett, 2005). The low scores demonstrate that sub-question generation is more challenging than a simple paraphrasing task. For the other models, we explore two scenarios with different inputs: (1) sub-LF p only and (2) a concatenation of p and the corresponding templated sub-question q^t. We report BLEU scores based on n-grams overlap and BERTScores measuring semantic similarity. The results in Table 8 suggest that: (1) Using BART-large as the generation model achieves the best performance and (2) incorporating the templated sub-questions into the model input can improve performance on all baselines, which makes sense because some tokens in q^t can be directly copied into the output question.

Furthermore, we use the best-performing model (i.e. BART-large with both p and q^t as the input) in Table 8 as the basic setting to explore the modeling of different contexts including the complex question Q and the history of templated sub-questions h_{q^t}. As shown in Table 9, we find that (1) adding context into the model's input can obtain higher metric scores, which suggests that context can help

[4]https://www.kaggle.com/c/quora-question-pairs

Generation Models	BLEU-2	BLEU-4	BERTScore
BART-paraphrase	10.6	2.7	88.0
Seq2Seq(LSTM)	17.8	6.4	90.8
Seq2Seq(LSTM)^t	18.7	6.7	91.3
Transformer	21.1	8.4	91.7
Transformer^t	23.4	9.1	92.6
BART-base	30.7	15.0	93.8
BART-base^t	32.0	15.9	94.1
BART-large	31.5	15.4	94.0
BART-large^t	**32.4**	**16.2**	**94.2**

Table 8: Question generation performance of different models. t denotes that the input incorporates templated sub-question, as well as the current sub-logical form.

Context	BLEU-2	BLEU-4	BERTScore
BART-large^t			
w/o Context	32.4	16.2	94.2
+ h_{qt}	33.3	16.5	94.6
+ Q	33.4	16.6	94.6
+ Q & h_{qt}	**34.1**	**17.1**	**94.8**

Table 9: Comparison of question generation performance when considering different contexts in the input.

in a dialogue. (2) Those settings that incorporate the original complex question Q generally perform better than the others, since the complex question contains the semantics of the sub-question to be generated. (3) BART-large with the input containing both Q and the history of templated sub-questions achieves the best performance. We also tried incorporating the history of sub-LFs h_{lf}, but it does not help further improve the performance.

Because automatic metrics like BLEU scores do not necessarily paint a full picture of the model performance, we manually check 100 generated questions. They are indeed of high quality and semantically similar to the human-written ones, see details in the second part of Appendix C.

4.3 Simulation

In this section, we demonstrate that our framework can pair with other KBQA parsers and use simulated user feedback to correct their errors. To simulate a dialogue, we develop a pipeline: (1) automatically translate a parser's predicted LFs into natural questions using the sub-question generation model equipped with the best-performing setting in Table 9, (2) use oracle error detection and train a generator to simulate a human user's corrections for these dialogues. This generator is a BART-large model that leverages the complex question and templated sub-questions as input to generate human feedback, (3) correct erroneous parses using the previously trained parse correction model under the best-performing setting in Table 7.

	BART-large	QGG
EM	60.9	-
EM*	**75.1**	-
F1	65.8	49.0
F1*	**75.7**	**56.5**

Attempt	EM	F1
BART-large		
1	75.1	75.7
2	78.7	79.9
3	**79.0**	**80.1**

Table 10: The left table shows the performance of two types of semantic parsers after correction through simulation process, BART-large and QGG. * denotes results after correction. The right table shows BART-large's performance after multiple attempts of correction.

We conduct simulation experiments on BART-large (Lewis et al., 2020) and QGG (Lan and Jiang, 2020b) respectively from two mainstream methodologies for KBQA as mentioned. We report both F1 and EM for BART-large before and after the correction process using the simulation pipeline. For QGG, since its generated query graphs do not take exactly the same format as SPARQL queries, we report F1 score only. As shown in the left part of Table 10, BART-large achieves a 14.2 EM and 9.9 F1 score gain after correction. Meanwhile, the correction process brings 7.5 F1 score improvement for QGG. The results show that INSPIRED can help train effective sub-question generation and parse correction models, which makes our framework applicable to KBQA parsers beyond the one used for constructing INSPIRED. Simulating user feedback makes it easy and far less costly to understand the potential of any base parser (as long as it outputs LFs) under our framework.

Moreover, we expand the simulation experiment to include multiple attempts of correction to simulate situations in which the model does not repair the parse correctly on the first attempt. We use the same human feedback generator to decode several of the highest scoring sequences as candidates for different attempts at correction. We evaluate this strategy after a maximum of three attempts.

Given that sequences decoded by plain beam search (Sutskever et al., 2014) often differ only slightly from each other, we adopt diverse beam search (Vijayakumar et al., 2018) instead to decode more diverse feedback. As shown in the right part of Table 10, F1 scores are up to 80.1 after three attempts of correction. We expect CBR-KBQA (the SOTA model mentioned earlier) to do even better given the advantages it has over plain seq2seq models. For example, their retrieve module can alleviate errors caused by sparse predicates. We envision the combination of our framework and theirs as interesting future work.

5 Related Work

Conversational Semantic Parsing. Conversational semantic parsing (CSP) is the task of converting a sequence of natural language utterances into LFs through conversational interactions. It has been studied in task-oriented dialogues, question answering and text-to-SQL. In task-oriented systems, datasets like MWoZ (Budzianowski et al., 2018; Eric et al., 2020) and SMCalFlow (Andreas et al., 2020) help users with a specific task (e.g., booking a hotel). CSQA (Saha et al., 2018), and CoQA (Reddy et al., 2019) are built for conversational systems to answer inter-related, simple questions. Meanwhile, ATIS (Hemphill et al., 1990; Dahl et al., 1994), SPARC (Yu et al., 2019) and CoSQL (Yu et al., 2020) are constructed for conversational text-to-SQL tasks. Our work shares a similar objective, i.e., how to represent natural language utterances while considering the multi-turn dynamics of the dialogue. We differ from them in that our task aims at soliciting and applying human feedback to correct generated initial parses.

Interactive Semantic Parsing. Multiple works have studied involving human feedback in the parsing process. Gur et al. (2018) ask multiple choice questions about a limited set of predefined errors. Yao et al. (2019b) ask yes/no questions about the presence of SQL components when generating one component at a time. Elgohary et al. (2020) introduce SPLASH, a dataset for correcting parses in text-to-SQL with free-form natural language feedback. Elgohary et al. (2021) propose NL-EDIT, converting feedback in SPLASH to a series of edits that can be deterministically applied to the initial parse. Su et al. (2018) and Elgohary et al. (2020) observe that most mistakes made by neural text-to-SQL parsers are minor and are thus resolved by changing a word or two. In contrast, parse errors in KBQA are often more difficult to resolve in fine-grained edits. Thus our framework allows users to provide feedback step by step. Unlike SPLASH that makes one-time correction to the entire initial parse, we propose to break down the parse into a sequence of sub-components and use the step-by-step feedback to correct corresponding one, reducing the burden of parse correction and increasing the likelihood of an accurate parse.

Question Decomposition. Question decomposition has been successfully used in complex QA. Iyyer et al. (2016) propose to answer questions based on tables by decomposing them into inter-related simple questions. Talmor and Berant (2018b) and Min et al. (2019) train a model directly to produce sub-questions using question spans. Recent works (Wang et al., 2020; Wolfson et al., 2020) introduce explicit annotation for the decomposition of multi-hop questions into a series of atomic operations. Wolfson et al. (2020) construct the BREAK dataset and propose QDMR, where questions are decomposed into a series of simpler atomic textual steps. QDMR is an intermediate representation of natural language and LFs, and is not executable on knowledge bases. In our work, we decompose the LF of the complex question into sub-queries, which can be directly executed on KB and retrieve answers. Moreover, we use decomposition to correct the initial parse at a finer-grained level.

6 Conclusion and Future Work

In this work, we propose an interactive semantic parsing framework and instantiate it with KBQA. Using this framework, we crowdsource a novel dataset, dubbed INSPIRED, and experimentally show that it can greatly increase the parse accuracy of an initial parser. Moreover, we design a simulation pipeline to explore the potential of our framework for a variety of semantic parsers, without further annotation effort. The performance improvement shows interactive semantic parsing is promising for further improving KBQA in general.

The INSPIRED dataset and experiments described here provide a foundation for many directions of future work. For example, this could take the shape of gains in accuracy as well as improvements to the correction strategy through decomposition. The simulation pipeline provided can also be used for further experimentation.

We plan to conduct a user study in which our framework is utilized by human users to query a knowledge base, in order to validate its viability for real use. At the moment, users are required to insert, delete, or replace whole sub-questions. A useful addition would be *modification* of text spans within a given sub-question, which requires a more fine-grained approach to connect SPARQL query components to natural language. Other complementary work could include handling errors introduced by named entity recognition and linking. Lastly, applying our framework to other query languages like SQL could be an exciting expansion.

7 Ethical Considerations

IRB Approval. Prior to collection of the INSPIRED dataset, we obtain IRB (Institutional Review Board) approval at our institution. This data collection is considered Exempt Research, meaning that our human subjects are presented with no greater than minimal risk by their participation. Participants' personal information is not collected, aside from minimal demographic information including their native language, which is used to ensure native-speaker level proficiency in the dataset. No identifying information is included. Further, all participants are required to read and agree to an *informed consent* form before proceeding with the task. AMT automatically anonymizes crowdworkers' identities as well.

Compensation to Crowdworkers. In order to ensure both quality data collection and fair treatment of our crowdworkers, we carefully review our payment plan for the AMT task. After a pilot study we gauge the average amount of time we expect a task to require and adjust the payment amount per task according to the minimum wage amount in our state, resulting in a 70 cent payment per task. Further, we ensure compensation for the time spent on the tutorial and qualification task by awarding $10 bonuses after completion of their first 10 tasks. They also receive $10 bonuses upon every 100 tasks they complete. In total, the cost of creating the INSPIRED dataset is approximately $13,300.

References

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, et al. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 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.

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

Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019. Uhop: An unrestricted-hop relation extraction framework for knowledge-based question answering. 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)*, pages 345–356.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Ahmed Elgohary, Ahmed Hassan Awadallah, et al. 2020. Speak to your parser: Interactive text-to-sql with natural language feedback. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2065–2077.

Ahmed Elgohary, Christopher Meek, Matthew Richardson, Adam Fourney, Gonzalo Ramos, and Ahmed Hassan Awadallah. 2021. NL-EDIT: Correcting semantic parse errors through natural language interaction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5599–5610, Online. Association for Computational Linguistics.

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 12th Language Resources and Evaluation Conference*, pages 422–428.

Sebastian Gehrmann, Tosin P Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna-Adriana Clinciu, Dipanjan Das, Kaustubh D Dhole, et al. 2021. The gem benchmark: Natural language generation, its evaluation and metrics.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.

Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. 2018. Dialsql: Dialogue based structured query generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1339–1349.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 553–561.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2337–2342.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2016. Answering complicated question intents expressed in decomposed question sequences. *arXiv preprint arXiv:1611.01242*.

Ravikumar Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical nlg framework for aggregated planning and realization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1406–1415.

K. Kukich. 1983. Design of a knowledge-based report generator. In *ACL*.

Yunshi Lan and Jing Jiang. 2020a. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974.

Yunshi Lan and Jing Jiang. 2020b. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

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

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Kathleen R McKeown. 1985. Discourse strategies for generating natural-language text. *Artificial intelligence*, 27(1):1–41.

Susan W McRoy, Songsak Channarukul, and Syed S Ali. 2000. Yag: A template-based generator for real-time systems. In *INLG'2000 Proceedings of the First International Conference on Natural Language Generation*, pages 264–267.

Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. Parlai: A dialog research software platform. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 79–84.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yu Su, Ahmed Hassan Awadallah, Miaosen Wang, and Ryen W White. 2018. Natural language interfaces with fine-grained user interaction: A case study on web apis. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 855–864.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. 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 2380–2390.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Alon Talmor and Jonathan Berant. 2018a. Repartitioning of the complexwebquestions dataset. *arXiv preprint arXiv:1807.09623*.

Alon Talmor and Jonathan Berant. 2018b. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search: Decoding diverse solutions from neural sequence models.

Ran Wang, Kun Tao, Dingjie Song, Zhilong Zhang, Xiao Ma, Xi'ao Su, and Xinyu Dai. 2020. R3: A reading comprehension benchmark requiring reasoning processes. *arXiv preprint arXiv:2004.01251*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

Ziyu Yao, Xiujun Li, Jianfeng Gao, Brian Sadler, and Huan Sun. 2019a. Interactive semantic parsing for if-then recipes via hierarchical reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ziyu Yao, Yu Su, Huan Sun, and Wen-tau Yih. 2019b. Model-based interactive semantic parsing: A unified framework and a text-to-sql case study. 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 5447–5458.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. 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 1321–1331.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. 2020. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. In *2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pages 1962–1979. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, et al. 2019. Sparc: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. Paws: Paraphrase adversaries from word scrambling. 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)*, pages 1298–1308.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

A Dataset Creation Details

The creation of the INSPIRED dataset requires careful selection of questions, design of a decom-

Question
What is the official language of the country that contains Al Sharqia Governorate?
SPARQL Query
<sparql-header-1> ?c ns:location.country.administrative_divisions #entity1# . ?c ns:location.country.official_language ?x .
Answer
Modern Standard Arabic

Table 11: Example question from the CWQ dataset. The entity "Al Sharqia Governorate" is replaced with "#entity1#". Entities are delexicalized in order to increase generalizability across questions in training.

positional approach, and a translation strategy between logical forms and human-readable language. Further, we carefully design a crowdsourcing task to gather more natural-sounding questions to enhance the quality and versatility of our framework.

A.1 Forming Dialogues from CWQ

We utilize the ComplexWebQuestions 1.1 (CWQ) dataset (Talmor and Berant, 2018a,b), as this is a common dataset used for complex question-answering over knowledge bases. This dataset is formed by combining questions from the WebQuestionsSP dataset (Yih et al., 2016) to form multi-hop complex questions, meaning that they require more than one step to answer. Each question has an associated SPARQL query that functions as a meaning representation of the question. Table 11 shows an example of a complex question, its associated SPARQL query, and its answer.

We envision that a human user will ask a complex question, the system will predict a SPARQL query for that question, decompose it into pieces, translate those pieces into English to show to the user to solicit feedback. The system will then use that feedback to correct the initial parse, if necessary. Figure 1 shows illustrations of this process.

In order to model this type of dialogue, we utilize a transformer-based seq2seq model (Vaswani et al., 2017) to predict a SPARQL query for each complex question and decompose the predicted and gold query into pieces, then use these pieces as editable chunks which can be deleted, replaced, or inserted to transform the predicted query into the gold. This process is the framework around which each dialogue is constructed. We translate each step from SPARQL into English to be comprehensible to a human user, thus resulting in dialogues like the one shown in Figure 1, all stemming from questions that occur in the CWQ dataset. Note that the parser used for this purpose is not state-of-the-art, as part of the goal is to have a broad coverage of error types for correction.

A.2 Translation of SPARQL Using Templates

As this dataset leverages SPARQL queries, we then develop a strategy for how to represent these queries in a more comprehensible form that humans can understand. Thus we create a template corpus and develop a rule-based translation method to do so. The corpus consists of 772 different predicates that appear in the CWQ dataset and translations of each into a basic template that conveys the content. The strategy of using templates to make content more human-friendly has a long history, both utilizing handcrafted templates (Kukich, 1983; McKeown, 1985; McRoy et al., 2000) and rule-based template formation (Angeli et al., 2010; Kondadadi et al., 2013). We use a blend of both approaches to create templates to represent logical forms in a way that is understandable to our crowdworkers. As can be seen in Table 11, SPARQL queries contain predicates that appear in the form of triples with each component separated by periods, such as *location.country.administrative_divisions* and *location.country.official_language*. These triples consist of a domain (*location*), a type (*country*) that represents a class within the domain, and a property (*administrative_divisions* and *official_language*) that specify more granular information. These predicates represent content information about the question and can appear in multiple, different questions. For example, the *location.country.administrative_divisions* predicate maps to the template *the country/countries that contain(s) <PH>*, where <PH> ("placeholder") gets replaced with a specific entity.

In the parsing process, we delexicalize these specific entities in order to make questions more generalizable and reduce noise during training. For example, in the SPARQL query in Table 11, the replacement token *#entity1#* appears, which we replace with *Al Sharqia Governorate* when the template is invoked.

The remaining components of the SPARQL query specify the question type and any additional components, which we leverage to transform the template into a full sentence. The components will

Composition	
Question	What is the mascot of the team that has Nicholas S. Zeppos as its leader?
SPARQL	<sparql-header-1> ?c ns:organization. organization.leadership ?k . ?k ns:organization.leadership.person #entity1# . ?c ns:education.educational _institution.mascot ?x .
Templates	1. the organization whose leadership includes a person named <PH> 2. the educational institution with the mascot <PH>
Translation	1. **What is/are** the organization whose leadership includes a person named **Nicholas S. Zeppos**? 2. **That entity is/are** the educational institution with the mascot **what**?
Conjunction	
Question	What country with the capital of Hagåtña is where Sam Shepard lives?
SPARQL	<sparql-header-2> #entity1# ns:people.person.places_lived ?y . ?y ns:people.place_lived.location ?x . ?x ns:location.country.capital #entity2# .
Templates	1. the person(s) who lived in <PH> 2. the location with the capital city named <PH>
Translation	1. **Sam Shepard is/are** the person(s) who lived in **what**? 2.**Of which, what is/are** the location with the capital city named **Hagåtña?**
Comparative	
Question	What country is in the Caribbean with a country calling code higher than 590?
SPARQL	<sparql-header-2> #entity1# ns:location.location.contains ?x . ?x ns:common.topic.notable_types #entity2# . ?x ns:location.country.calling_code ?num . filter (xsd:integer (?num) > 590) .
Templates	1. the location(s) containing <PH> (<RSTR>) 2. the country/countries whose calling code is/are <PH>
Translation	1. **Caribbean is/are** the location(s) containing **what (country)**? 2. **Of which, what is/are** the country/ countries whose calling code is/are **greater than 590**?
Superlative	
Question	Which pro athlete started his career earliest and was drafted by the Cleveland Browns?
SPARQL	<sparql-header-2> #entity1#ns:sports. professional_sports_team.draft_picks ?y . ?y ns:sports.sports_league_draft_pick. player ?x . ?x ns:sports.pro_athlete. career_start ?num . } order by ?num limit 1
Templates	1. the team(s) that drafted the athlete(s) <PH> 2. the pro athlete(s) who started their career(s) in <NUM>
Translation	1. **Cleveland Browns is/are** the team(s) that drafted the athlete(s) **what**? 2. **These entities are** the pro athlete(s) who started their career(s) in **what**? 3. **Of these, which is the entity associated with the earliest date?**

Table 12: Question types from the CWQ dataset and the translation process to templated sub-questions.

be discussed more fully in A.3. Thus, this particular SPARQL query translates to the following sub-questions:

1. *What is/are the country/countries that contain(s) [Al Sharqia Governorate]?*

   ANSWER: Egypt

2. *That entity is/are the country/countries whose official language is what?*

   ANSWER: Modern Standard Arabic

A.3 Question Types

Each of the questions in CWQ can be categorized into one of four major reasoning types: composition, conjunction, comparative, and superlative (Talmor and Berant, 2018b). Each type can be identified by the SPARQL query and translated accordingly. Table 12 shows the translation process of the four types with examples of each. The general strategy is to append content to the beginning of the template and replace the <PH> token to form a complete question and express the appropriate question type. As seen in Table 12, this is quite straightforward for composition- and conjunction-type questions.

Composition questions are composed of two simple questions, where the answer to the first is used to form the second question. As an example, in order to answer the question *What is the mascot of the team that has Nicholas S. Zeppos as its leader?*, one must first answer *In which organization is Nicholas S. Zeppos a leader?* to have all the content necessary to answer *What is the mascot of that organization?*. To translate these question types to templated sub-questions, we simply append *What is/are* before the first template and insert the named entity where the <PH> token appears in the template. Then, *That entity is/are* is appended to the beginning of the second template and *what* replaces the placeholder. Note that these positions can be reversed depending on what content is provided in the question. For example, a question could be either of the two options, depending on the goal of the target question:

1. ***What is/are** the organization whose leadership includes a person named **Nicholas S. Zeppos**?*

2. ***Vanderbilt University is/are** the organization whose leadership includes a person named **what**?*

Conjunction questions follow a very similar process, though because their goal is to find the intersection of two categories, the first question returns a list of answers. To account for this, we simply append *Of which* to the second question before following the same set of rules as the composition questions.

Comparative questions generally have a comparative operator (<, >) and a number contained in their SPARQL query, which we translate simply to *less than X* or *greater than X*, as appropriate. Note that the comparative example in Table 12 contains a "restriction predicate", marked by the <RSTR> token. This will be discussed in Section A.3.1.

Superlative questions require a slightly more complicated strategy. The first sub-question of a superlative type question always generates a list of answer options, while the second sub-question must pair those answer options with numerical information, such as dates or integers. Then, these numbers are ordered, either from smallest-to-largest or vice versa, and the first is returned as the final answer. To account for this, we append *These entities are* to front of the second template, to make it clear that multiple entities are involved, and return a paired list of entities and their corresponding values as an answer. Then we append a third sub-question that specifies how the questions are sorted and returns a single answer.

A.3.1 Logical Form Features

Within the four main types of questions (composition, conjunction, comparative, and superlative), there are a variety of features that appear. These features include filters, restriction predicates, and union predicates.

Filters act to restrict a list of entities in some fashion by assigning numerical boundaries. An example of this can be seen in Table 12 in the comparative question's SPARQL query, starting with the word *filter*. This sequence limits the list of entities by ones whose calling codes are larger than 590.

Restriction predicates can appear as auxiliary pieces to regular predicates and typically provide categorical information about an entity. For example, in Table 12, the comparative-type question *What country is in the Caribbean with a country calling code higher than 590?* has two entities in its SPARQL query, though *Caribbean* is the only entity that seems to appear in the original question. The two main predicates are *location.location.contains* and *location.country.calling_code*, but a third predicate, *common.topic.notable_types* appears in between them. This predicate acts as a restriction upon the first main predicate; in this case *#entity2#* corresponds to *country* and restricts the locations that can appear as answers to the category of countries.

Composition	
Question	Who is both a member of the Kennedy family and the Order of the British Empire?
SPARQL	**filter (?x != #entity1#) { # parents #entity2# ns:people.person.parents ?x . } union { # children #entity3# ns:people.person.children ?x . } union { # siblings #entity4# ns:people.person.sibling_s ?y . ?y ns:people.sibling_relationship.sibling ?x . } union { #spouse #entity5# ns:people.person.spouse_s ?y . ?y ns:people.marriage.spouse ?x . ?y ns:people.marriage.type_of_union #entity6# . filter (not exists { ?y ns:people.marriage.to []}) }** ?x ns:royalty.chivalric_order_member.belongs_to_order ?c . ?c ns:royalty.chivalric_order_membership.order #entity7# .
Templates	1. the family of <PH> 2. the member(s) of the order of <PH>
Translation	1. **Who is/was** the family of **John F. Kennedy**? 2. **Of which, what is/are** the member(s) of the order of **Order of the British Empire**?

Table 13: Example of a question whose SPARQL query includes a union predicate.

Because restriction predicates are not standalone pieces that could be translated into their own sub-questions, we develop a strategy for incorporating them into the templates of the predicates they restrict. First, we create a corpus of "mini-templates" that correspond to all the restriction predicates that could appear. Much of the time, these mini-templates simply place the entity (like *country* in the previous example) into parentheses, though in some cases they situate the entity into a prepositional phrase.

Meanwhile, the main template corpus has tokens in place to define where the mini-template should be placed in the main template. One can see in the comparative example of Table 12 that there is an <RSTR> token in the template of the first sub-question. Every main template that can appear with a restriction predicate has this token in its template; though it needs not always appear with one. Consequently, if the restriction token does not get replaced, it simply gets deleted. If the *location.location.contains* predicate appeared without a restriction predicate, it would simply read *Caribbean is/are the location(s) containing what?*

Union predicates are a bit of a misnomer, as they are actually a group of predicates that function as though they are a single predicate, and thus correspond to a single template. In Table 13, one can see that the SPARQL query is quite long, with all of the content in bold corresponding to the first sub-question and the remainder corresponding to the second. Within this first sub-LF, there are several predicates that are joined together by *} union {*. Collectively, these templates encompass the concept of *family* by defining all the various relationship roles that are involved in that concept. Theoretically, we could enumerate all of these in template form, separated by *or* (*the brother of John F. Kennedy* ***or*** *the mother of John F. Kennedy* ***or*** *the child of John F. Kennedy...*) but this seems to be an unnecessarily complicated and inconcise way of representing these. Instead, we enumerate the various types of union predicates that could appear and create a small corpus of templates that express the overall concept represented by each collection of predicates, thus crowdworkers will see questions with this feature in the same format as a regular question.

A.4 Crowdsourced Data Collection

As mentioned in Section 2.2, the crowdsourcing task for this dataset is primarily a paraphrasing task in which crowdworkers work through a structured dialogue, rephrasing templated sub-questions at each step.

Each task takes the form of a dialogue involving three entities: the "user", which is an automated dialogue partner, an automated "director" that guides the dialogue and provides detailed instructions, and the "agent", which is the role performed by the crowdworker. Upon entering a task, the worker is shown the "target question", or the original question from CWQ, and asked if the question was sensible to them. If so, they are asked to rephrase it using different language. If not, they proceed with the dialogue in the hopes that the decomposition process will make the meaning of the question clear. In these cases, the crowdworker is asked to rephrase the target question at the end of the dialogue. This process is included to encourage better understanding of the target question and to help us recognize confusing questions in the original dataset and replace them with higher-quality questions when appropriate.

Next, the target question is automatically decomposed into templated sub-questions which are displayed to the worker, who rephrases them into English. These rephrased questions are sent to the automated user, who provides corrections as necessary. The worker rephrases any new questions and the edits are automatically made. At the end of the dialogue, the worker is asked for any feedback regarding the dialogue. This feedback is later used to make corrections and flag any problems that might have arisen. Screenshots of the dialogue interface can be seen in Figure 3.

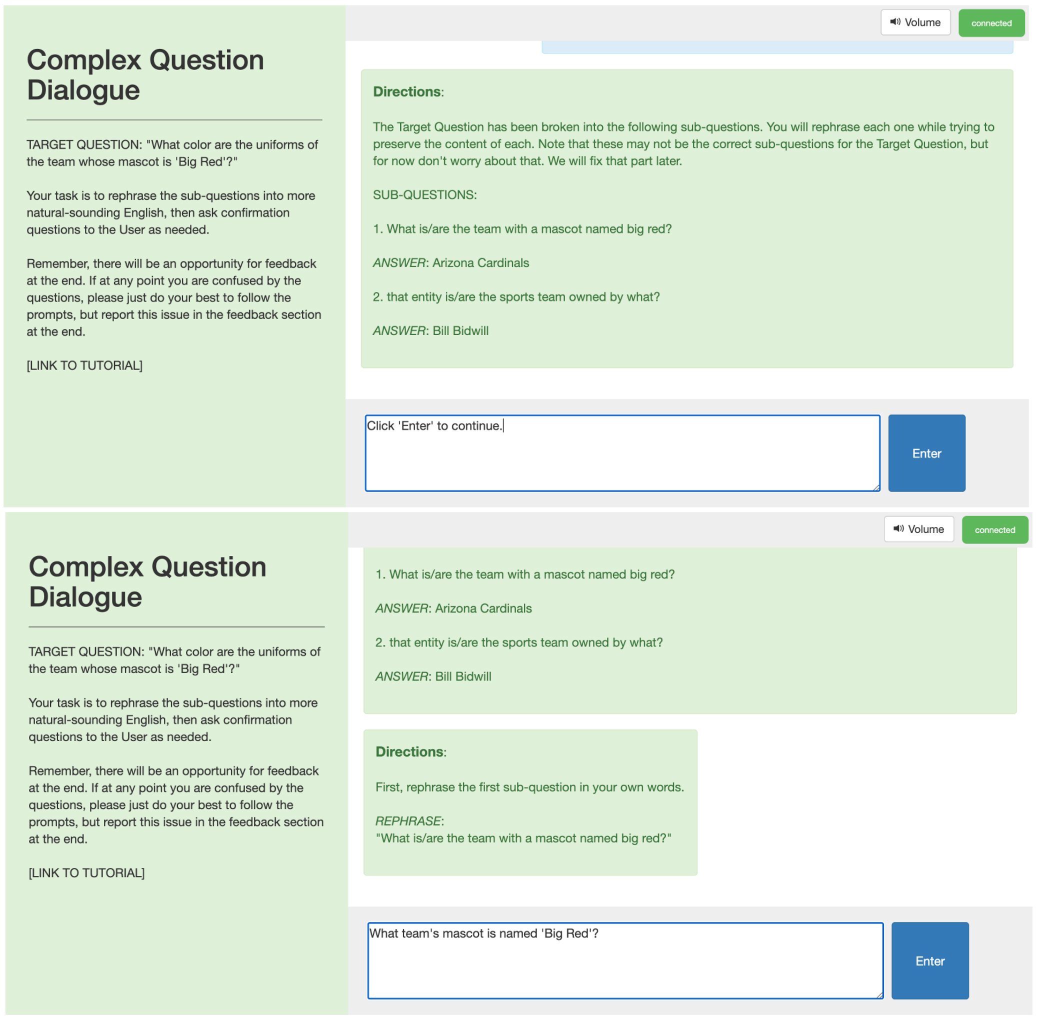
B Dataset Analysis

B.1 Cleaning the Dataset

As mentioned in Section 3, we employ a semi-automatic data cleaning method to reduce the error rate in the INSPIRED dataset. Because data cleaning can be an expensive and time-consuming process, the goal is to develop a method that would reduce the number of items in the dataset that need to be manually reviewed. Thus we use an automatic method to identify a small subset of the entire dataset that contain as many errors as possible to then manually review. To this end, we utilize a pre-trained sequence-to-sequence model that employs the idea of *cycle consistency* (Zhu et al., 2017), to identify poor paraphrases by retrieving meaning representations (MRs) from questions rephrased by the workers. Then these MRs are used to compare against the original MRs and evaluated for similarity.

In order to evaluate the effectiveness of the strategy, a random 5% subset of the entire dataset is selected for annotation, using a binary classification of whether or not the rephrased question was an accurate paraphrase of the original templated question (and by extension, its original logical form). This annotation effort revealed that 4.4% of the rephrased questions contain errors, which we expect is representative of the entire dataset.

We then fine-tune Hugging Face's implementation of T5 in a seq2seq model to generate MRs, in this case templated sub-questions, to compare to the original MRs (Wolf et al., 2020; Raffel et al., 2020). These pairs of MRs then need to be sorted in a ranked list that filters paraphrases that are more likely to contain errors to the top of the list. This allows us to use a *precision at K* measure, which, given a rank K, the precision is calculated over the set of retrieved items with a rank of K or less. For the annotated test set, K equals 75, the number of observed errors. After ranking the list, we can


Complex Question Dialogue

TARGET QUESTION: "What color are the uniforms of the team whose mascot is 'Big Red'?"

Your task is to rephrase the sub-questions into more natural-sounding English, then ask confirmation questions to the User as needed.

Remember, there will be an opportunity for feedback at the end. If at any point you are confused by the questions, please just do your best to follow the prompts, but report this issue in the feedback section at the end.

[LINK TO TUTORIAL]

Directions:

The Target Question has been broken into the following sub-questions. You will rephrase each one while trying to preserve the content of each. Note that these may not be the correct sub-questions for the Target Question, but for now don't worry about that. We will fix that part later.

SUB-QUESTIONS:

1. What is/are the team with a mascot named big red?

ANSWER: Arizona Cardinals

2. that entity is/are the sports team owned by what?

ANSWER: Bill Bidwill

Click 'Enter' to continue.

Complex Question Dialogue

TARGET QUESTION: "What color are the uniforms of the team whose mascot is 'Big Red'?"

Your task is to rephrase the sub-questions into more natural-sounding English, then ask confirmation questions to the User as needed.

Remember, there will be an opportunity for feedback at the end. If at any point you are confused by the questions, please just do your best to follow the prompts, but report this issue in the feedback section at the end.

[LINK TO TUTORIAL]

1. What is/are the team with a mascot named big red?

ANSWER: Arizona Cardinals

2. that entity is/are the sports team owned by what?

ANSWER: Bill Bidwill

Directions:

First, rephrase the first sub-question in your own words.

REPHRASE:
"What is/are the team with a mascot named big red?"

What team's mascot is named 'Big Red'?

Figure 3: Data collection interface on AMT, using the ParlAI framework (Miller et al., 2017).

evaluate the quality of the method by checking the top K data points and checking to find how many errors appear in that set, compared to a random baseline of 4.4% (the observed error rate), or about 3 errors.

We employ two ranking methods to sort the pairs. First, we calculate the negative log-likelihood of the target MRs relative to the model and then do the same for the generated MRs.

$$\mathcal{S}(y) = -\sum_{y_i \in Y} \log p(y_i | y_{<i}, x; \theta) \tag{1}$$

$$y = \langle y_1, ..., y_{|y|} \rangle$$
$$y_{<i} = \langle y_1, ..., y_{i-1} \rangle$$

$S(y)$ refers to the score of a given output sequence y, which is the sum of the negative log-likelihood of each y_i given the sequence of y tokens that come before. θ refers to the model parameters.

Once the negative log-likelihoods are determined for each candidate y, the best candidate is determined based on the lowest score.

$$y* = argmin(S(y|x)) \tag{2}$$

Here, $y*$ refers to the best generated output sequence, and x is a given input sequence. A score for output sequence $y*$ is determined, as well as a score for the target sequence t.

$$D = |S(y*) - S(t)| \tag{3}$$

While these two scores are comparable to each other, they are not comparable across other item pairs. In order to assign a ranking for every item in the dataset, we calculate the difference D between the negative log-likelihoods of the target MR and generated MR for each question in the dataset and sort them based on the largest difference score, as shown in Equation 3.

Second, we calculate an edit distance score between the target MR and generated MR and sort based on the largest score. If the model has predicted an MR that is substantially far from the target MR in its phrasing, it likely has a different meaning.

Using the first ranking method, 17.3% of the errors are recovered, while the second recovers 32% of the errors. However, because the two ranking methods appear to be identifying different errors with little overlap, both are used to identify the final set of questions for manual review, drawing from the methods equally.

Then the method is applied to the entire INSPIRED dataset, using cross-validation with a series of 90% training, 10% testing splits to generate MRs for every rephrased question. Then, because the annotated dataset has a 4.4% error rate which we expect to be representative of all the data, the top-ranked 4.4% of data is selected for manual review. This review results in 17.7% of items being revised, meaning that authors change the rephrasing to more accurately reflect the original meaning.

B.2 GEM Metrics

	Template Corpus	Rephrased Corpus
Unigrams		
Vocab Size	8,465	9,864
Distinct	0.012	0.022
Unique	1,003	1,258
Entropy	6.532	8.090
Bigrams		
Vocab Size	21,072	44,085
Distinct	0.031	0.109
Unique	2,949	8,723
Entropy	8.976	12.484
Cond Entropy	2.295	3.918
Trigrams		
Vocab Size	31,838	81,479
Distinct	0.050	0.224
Unique	5,332	20,971
Entropy	10.291	14.529
Cond Entropy	1.250	1.986

Table 14: GEM n-gram metrics for the template corpus and rephrased question corpus.

Table 14 shows the N-gram statistics of all the templates in the dataset (template corpus) and all the rephrased questions (rephrased corpus). These metrics are calculated using the GEM evaluation scripts (Gehrmann et al., 2021). In this table, *Vocab Size* refers to the total number of distinct N-grams, while *Distinct* refers to the ratio of distinct N-grams divided by the total number of N-grams in the dataset. *Unique* specifies the number of N-grams that occur only once in the dataset, *Entropy* is the Shannon entropy over N-grams, and *Cond(itional) Entropy* is the entropy conditioned on N_{-1}-grams.

Lexical Relationship	**Percentage(%)**
Lexical Match	58
Synonymy	31
Hypernymy	5
Hyponymy	20

Table 15: Lexical analysis of 100 randomly sampled sub-questions and their templates. Note that *Lexical Match* refers to the percentage of words in all sub-questions that appear in their corresponding templated question.

B.3 Lexical Analysis

In order to better understand the methods by which crowdworkers rephrased templates, 100 randomly selected sub-questions are studied in terms of the lexical relationships between the template and rephrased versions. Table 15 shows the results of this analysis. "Lexical match" refers to the average proportion of words in the rephrased version that also appear in the template, relative to the total number of words in the rephrased version. Synonymy, hypernymy, and hyponymy refer to the number of questions in the 100 selected items that contain an instance of one of these lexical relations. It is clear, therefore, that crowdworkers are using these strategies in their rephrasings of the templates, in addition to simply changing word order. On average, a bit less than half the words in a rephrased question are newly introduced by the crowdworker, and 56% of the time they are using synonmy, hypernymy, hyponymy, or some combination of these to rephrase the templated question.

B.4 Contextual Awareness

In a given dialogue, we provide answers to the sub-questions when possible, making the dialogue context-rich and providing the user with as much information as possible to help them understand the decomposition process of their original query.

This context-awareness can also be seen in the sub-question paraphrases. Our crowdworkers are encouraged to paraphrase questions in a manner that accounts for the overall context of the question, particularly with regard to named entities. For example, when a second sub-question references the answer of the first sub-question, we ask the

Sub-question predicate	Actual Context	Random Context
film.film_subject.films	Complex Question: **Who was** the wife of **the subject of the** film #entity#? Sub-Questions: *1. Who was the subject of the movie #entity#? 2. Who was that person married to?	Complex Question: Where did the topic **of the** film #entity# pass away at? Sub-Questions: *1. Who was the main focus in the movie called #entity#? 2. Where did this individual die?
influence.influence_node. influenced_by	Complex Question: Which peer of #entity# **inspired the work of** #entity#? Sub-Questions: *1. Who inspired the work of #entity#? 2. Of the above named people, which had a peer relationship with #entity#?	Complex Question: What person **who** influenced #entity# 's **work** was born on #entity#? Sub-Questions: *1. #entity# inspired which people's work? 2. Which of these people were born on #entity#?

Table 16: Examples of sub-questions in their actual context vs. a random context that utilizes the same predicate in its logical form. The sub-question was substituted for the one that used the same logical form (marked with *) in the random context when calculating ROUGE scores. Lexical overlap of the sub-question with each context is represented by bold text. Entities have been replaced with #entity# tokens in order to avoid disadvantaging the random context due to overlap in named entities.

Turkers to reference that entity without naming it explicitly, but also using a more specific phrase than *entity*. An example of this can be seen in Figure 1, where instead of directly incorporating the answer of the first question (*Egypt*) into the second question, they reference it using the phrase *the above-named nation*. The goal of this strategy is to create a dataset of dialogues that are context-aware and grounded, on which generation models can be trained to mirror this behavior. By using less specific phrases than entity names, our model is better able to generalize across examples during training.

However, one can envision that in a real-use situation, it might be more natural for a user to simply use *Egypt* instead of *the above-named nation* when correcting sub-question 2. While our current framework is not able to accommodate this behavior, a simple data augmentation procedure in which referring expressions are replaced with the named entities should allow our system to accommodate this. We leave this data augmentation for future work, but plan to implement it upon conducting a study with real users.

In order to demonstrate contextual awareness, Table 4 in Section 3 shows the average ROUGE-1 and ROUGE-2 scores of all sub-questions in their actual contexts compared with the same sub-questions in a randomly assigned context that utilizes the same sub-logical form. The higher scores for the actual context indicate that the wording of sub-questions reflect the context from which they are derived. Moreover, Table 16 shows examples of sub-question with these context comparisons.

	Conjunction	Composition	Comparative	Superlative	Total
Delete	49	94	9	3	155
Insert	1835	765	208	208	3016
Replace	2207	1757	341	393	4698
No Action	2172	2240	301	310	5023

Table 17: Distribution of error types within the sub-questions of the four main question types.

B.5 Error Characteristics of Initial Parser

It is important to note that our initial parser is purposefully not state-of-the-art, as we want to have a wide distribution of errors around which we could create dialogues. (See Section 4 for details about the initial parser.) Similar to other interactive semantic parsing work, we envision that the user will provide corrections to the sub-questions, though we at this stage require the user to use the three operations of deleting, replacing, or inserting a whole sub-question. Table 17 shows the distribution of sub-questions whose original complex question is of each of the four main types. Within these types, the distribution of edit operations per sub-question is shown. Though many of sub-questions do not need any edits, the *replace* operation is most frequent of edit operations, appearing in roughly 36.5% of each type, while *insert* is roughly 23.3% and *delete* is around 1.2%, with *no action* making up the remaining 39%. These distributions indicate the parser is more likely to predict something incorrect or leave out a sub-question, rather than predict a sub-question that is not present in the gold.

C Sub-Task Error Analyses

Parse Correction. We sample 100 erroneous predictions of BART-large under the best-performing setting in Table 7. In this analysis, it becomes clear

Sub-Q:
What tourist attractions are by the grand canyon?
Gold Sub-LF:
#entity1# ns:travel.tourist_attraction.near_travel_destination ?x .
Generated Sub-LF:
#entity1# ns:travel.travel_destination.tourist_attractions ?x .
Sub-Q:
What is that country's national anthem?
Gold Sub-LF:
?c ns:location.country.national_anthem **?y . ?y** ns:government.national_anthem_of_a_country.anthem ?x .
Generated Sub-LF:
?c ns:location.country.national_anthem ?x .

Table 18: Two error cases about wrongly generated predicates in an analysis of 100 generated sub logical forms. $?y$ in the logical form is an example of CVT node which connects two predicates that operate as a single, compound predicate.

Human-Written	*Which of the above named people did the voice of toki?*
Machine-Generated	*Which of these people played the role of toki?*
Error Explanation	Generated question does not specify that the role was a voice acting one.
Human-Written	*What famous person has addison's disease?*
Machine-Generated	*who has suffer from addison's disease?*
Error Explanation	Grammatical error
Human-Written	*What district does that politician represent?*
Machine-Generated	*What district does that person represent?*
Error Explanation	Generated question is slightly less specific

Table 19: Three instances of errors in an analysis of 100 generated sub-questions compared to human-written versions.

that longer, more complicated logical forms are more likely to be mispredicted. Only 21 of the errors involve single predicates, while 54 erroneous parses occur with *CVT (Compound Value Type)* predicates, which are essentially two predicates combined together via *CVT nodes* (for example *?y* in Table 18) that function as a single predicate. 13 errors occur on *restriction* predicates, which co-occur with single or *CVT* predicates to further limit the entity type. For example, predicates of the *location* domain might occur with a restriction that limits that predicate to locations of the type *country*. The remaining 12 errors all occur due to only partially generating a long logical form that contains filters. Details regarding restriction predicates and filters can be found in A.3.1.

Sub-Question Generation. We conduct an analysis on 100 randomly selected pairs of human-written question and machine-generated question that correspond to the same logical form. We first examine questions from the best-performing model in Table 9 according to BLEU scores and BERTScores, which use the current sub-logical

Better Model	Neither	Model 1	Model 2
Number	74	20	6
EXAMPLES			
Human-Written	who were walt disney's kids?	kevin hart went to what schools?	what is the name of the currency used in that country?
Model 1 (w/context)	who are the children of walt disney?	what **schools** did kevin hart go to?	what kind of currency do they use?
Model 2 (w/o context)	what are the names of walt disney's children?	what did kevin hart go to?	what is the currency used **in that country**?

Table 20: Comparison of 100 generated sub-questions from models with and without context in their inputs. The bolded text in columns 2 and 3 highlight what enhanced the quality of the generation in comparison to its counterpart. Model 1 refers to the model that used the complex question and previous templated questions as context (row 4 in Table 9) and Model 2 refers to the model that did not use context at all (row 1 in Table 9).

form, the current templated sub-question, the complex question and the history of templated sub-questions from previous steps as context. Questions in which the machine-generated and human-written versions exactly match each other were excluded. This analysis reveals that only three generated questions (3%) are of perceptibly worse quality than the human-written questions, as can be seen in Table 19. Further, there are four cases in which the human-written questions contain grammatical errors, whereas the machine-generated ones do not. An analysis of all generated questions which do not exactly match their human-written counterpart reveals that 64% of the generated questions are shorter in terms of number of words.

Because BLEU scores do not necessarily paint a full picture of the model performance, we then examine the generated responses from the model that produced the lowest BLEU scores, which is the model with no context. By examining the same 100 samples as in the previous analysis, we note twenty cases in which the best-performing model that leverages context better reflects that context in its rephrasing than the model that does not leverage context. There are, however, 6 cases in which the model without context does this better and in the remaining cases there is no discernible difference between the quality of the generations from the two models. Table 20 shows examples of each of these cases, for illustration.

While these results are based our observations and certainly require further future investigation and human annotation by people other than the authors, these preliminary results show that the generated questions can be more concise and of comparable quality.