Robust Training for Conversational Question Answering Models with Reinforced Reformulation Generation

Magdalena Kaiser  
Max Planck Institute for Informatics  
Saarland Informatics Campus  
Germany  
mkaiser@mpi-inf.mpg.de

Rishiraj Saha Roy  
Max Planck Institute for Informatics  
Saarland Informatics Campus  
Germany  
rishiraj@mpi-inf.mpg.de

Gerhard Weikum  
Max Planck Institute for Informatics  
Saarland Informatics Campus  
Germany  
weikum@mpi-inf.mpg.de

ABSTRACT

Models for conversational question answering (ConvQA) over knowledge graphs (KGs) are usually trained and tested on benchmarks of gold QA pairs. This implies that training is limited to surface forms seen in the respective datasets, and evaluation is on a small set of held-out questions. Through our proposed framework Reign, we take several steps to remedy this restricted learning setup. First, we systematically generate reformulations of training questions to increase robustness of models to surface form variations. This is a particularly challenging problem, given the incomplete nature of such questions. Second, we guide ConvQA models towards higher performance by feeding it only those reformulations that help improve their answering quality, using deep reinforcement learning. Third, we demonstrate the viability of training major model components on one benchmark and applying them zero-shot to another. Finally, for a rigorous evaluation of robustness for trained models, we use and release large numbers of diverse reformulations generated by prompting GPT for benchmark test sets (resulting in 20x increase in sizes). Our findings show that ConvQA models with robust training via reformulations, significantly outperform those with standard training from gold QA pairs only.

CCS CONCEPTS

• Information systems → Question answering

KEYWORDS

Question answering, Knowledge graphs, Conversations, Reformulations, Reinforcement learning

1 INTRODUCTION

Motivation. Answering questions about entities, powered by curated knowledge graphs (KGs) at the backend, is a vital component of Web search [7, 43, 64, 82]. Nowadays, users’ information needs are increasingly being expressed as a conversation, in a sequence of questions and answers: $(Q_1, A_1), \ldots$ over turns \{t\} [16, 53, 84]:

\[
\begin{align*}
Q_1 & : \text{What’s the 2022 LOTR TV series called?} \\
A_1 & : \text{The Rings of Power (TROP)} \\
Q_2 & : \text{TROP airing on?} \\
A_2 & : \text{Amazon Prime Video} \\
Q_3 & : \text{Which actor plays Isildur in the series?} \\
A_3 & : \text{Maxim Baldry} \\
Q_4 & : \text{And who is the Jackson trilogy?} \\
A_4 & : \text{Harry Sinclair} \\
Q_5 & : \text{When did the series start?} \\
& \ldots
\end{align*}
\]

A conversation over a KG contains a set of entities ("The Lord of the Rings: The Rings of Power", "Amazon Prime Video"), their relationships ("aired on"), and types ("TV series, video streaming service"). In ConvQA, users omit parts of the context in several follow-up turns $(Q_3 - Q_5)$, and use ad hoc style $(Q_2)$ [16, 17, 25, 34, 61, 67]. A part of the intent being left implicit, coupled with the use of informal language, make the answering of conversational questions more challenging than complete ones tackled in older and more established branches of QA [28, 64, 73, 76]. ConvQA has high contemporary interest [15, 33, 41, 58, 75], spurred on to a big extent by systems like ChatGPT that support a conversational interface.

Limitations of state-of-the-art. We quantify robustness in QA in terms of the number of distinct question formulations of a given intent, that a QA model can answer correctly: the higher this number, the more robust the model. Methods for conversational question answering (ConvQA) over KGs are usually trained and evaluated on benchmarks of gold-standard (question, answer) pairs [14, 25, 66, 67]. Such a paradigm limits robust learning by being restricted to question formulations roughly seen during training time. One approach in QA to demonstrate generalizability is to train and evaluate models on multiple benchmarks [33, 38, 48]. This only addresses the problem partially: the training and evaluation are still limited to surface forms seen in any of the benchmarks. A particular aspect of existing benchmarks, that is attributable to their construction choices via graph sampling [66] or crowdsourcing guidelines [12, 14], is that they often do not contain sloppy question formulations that could be asked by real users in the wild.

In the example conversation, $Q_4$ is phrased in a very casual way, asking for Isildur’s actor in the LOTR movie trilogy (Peter Jackson directed the LOTR movies). With this difficult input, the
QA system may give a wrong response. A seemingly natural approach to counter such effects would be to have the QA system automatically reformulate the question into a more complete version [3, 9, 10, 62, 75, 83], such as Which actor played the role of Isildur in the Lord of the Rings movie trilogy directed by Peter Jackson? – this kind of run-time question rewriting to a complete natural language form in a deployed system may sometimes work, but adds inference-time overhead and may not improve performance [30].

**Approach.** We take a different route: instead of reformulating a conversational question at inference time, we strengthen the training of the ConvQA model by exposing it upfront to a larger variety of intent-preserving surface forms for the same training data for the QA model for maximum performance improvement. This work calls for more robust training and evaluation of ConvQA models, our salient contributions being:

- A novel taxonomy of question reformulations for ConvQA over KGs, based on string edit distance;
- A reinforcement learning model with Deep Q-Networks, that systematically manipulates parts of a given conversational question based on string edit operations. For each category, we generate noisy supervision data to fine-tune an LLM, that then serves as our reformulation generator (RG, gray boxes). New lexico-syntactic forms in reformulations originate via use of a rich set of aliases in KGs, and world knowledge in LLMs.

QA system may give a wrong response. A seemingly natural approach to counter such effects would be to have the QA system automatically reformulate the question into a more complete version [3, 9, 10, 62, 75, 83], such as Which actor played the role of Isildur in the Lord of the Rings movie trilogy directed by Peter Jackson? – this kind of run-time question rewriting to a complete natural language form in a deployed system may sometimes work, but adds inference-time overhead and may not improve performance [30].

**Approach.** We take a different route: instead of reformulating a conversational question at inference time, we strengthen the training of the ConvQA model by exposing it upfront to a larger variety of intent-preserving surface forms for the same training data for the QA model for maximum performance improvement. This work calls for more robust training and evaluation of ConvQA models, our salient contributions being:

- A novel taxonomy of question reformulations for ConvQA over KGs, based on string edit distance;
- A reinforcement learning model with Deep Q-Networks, that systematically manipulates parts of a given conversational question based on string edit operations. For each category, we generate noisy supervision data to fine-tune an LLM, that then serves as our reformulation generator (RG, gray boxes). New lexico-syntactic forms in reformulations originate via use of a rich set of aliases in KGs, and world knowledge in LLMs.

Given that our generated instances are noisy, it is unlikely that for a given question, all categories of reformulations would improve the ConvQA model’s performance. As a result, for each question, we would like to judiciously select a few of these that are most beneficial. So we pass generated reformulations to the QA model as proxies to obtain human-like responses, and world knowledge in LLMs.

This entire framework, termed REIGN, (REInforced reformatioN Gereration) is the main contribution of this work.

**Evaluation.** To assess the benefits of REIGN, we perform experiments against two state-of-the-art baselines: CONQUER [35] based on reinforcement learning, and EXPLAINN [15] based on graph neural networks. Note that REIGN operates by model-aware training on top of these baselines. For test data, we leverage the generative ability of ChatGPT (GPT-3.5) as a proxy to obtain human-like reformulations at scale: each original question is augmented with 20 proxies against two state-of-the-art baselines: CONQUER [35] based on reinforcement learning, and EXPLAINN [15] based on graph neural networks. Note that REIGN operates by model-aware training on top of these baselines. For test data, we leverage the generative ability of ChatGPT (GPT-3.5) as a proxy to obtain human-like reformulations at scale: each original question is augmented with 20 proxies.

**Contributions.** This work calls for more robust training and evaluation of ConvQA models, our salient contributions being:

- A novel taxonomy of question reformulations for ConvQA over KGs, based on string edit distance;
- A reinforcement learning model with Deep Q-Networks, that selects helpful reformulations of conversational questions guided towards better QA performance;
- About 335k conversational question reformulations of test cases in two ConvQA benchmarks, suitable for rigorous evaluation of future models;
- The REIGN framework with reusable components that judiciously augments benchmark training tailored to specific ConvQA models. All code is at https://reign.mpi-inf.mpg.de.

### 2 CONCEPTS AND NOTATION

Salient notation is in Table 1 (some concepts introduced in Sec. 3).

**Knowledge graph.** A knowledge graph (KG) consists of a set of real-world objective facts. Examples of large curated KGs (equivalently, knowledge bases or KBs) include Wikidata [78], DBpedia [4], YAGO [69], or industrial ones (e.g., Google KG).

**Fact.** A KG fact is an SPO (subject, predicate, object) triple, where a subject is an entity (Lord of the Rings); an object is another entity (Maxim Baldry), a type (TV Series), or a literal (01 September 2002); and a predicate is a relationship (cast member) between the subject and the object. Compound facts involving more than two entities or literals are stored as a main triple and additional (predicate, object) pairs (referred to as “qualifiers” in Wikidata [78]). For example, the main triple (The Rings of Power, cast member, Maxim Baldry) has a qualifier (character role, Isildur).

---

**Figure 1: Performance-guided reformulation generation in REIGN, illustrated through our running example conversation.**
Table 1: Notation for concepts in REIGN.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C, t \in {1, 2, \ldots}$</td>
<td>Conversation, conversational turn</td>
</tr>
<tr>
<td>$Q = (q_1 \ldots q_n), A$</td>
<td>Question and its tokens, Answer</td>
</tr>
<tr>
<td>$Q_t, A_t$</td>
<td>Question and answer at turn $t$</td>
</tr>
<tr>
<td>${R_C^t_i}$</td>
<td>RCS-predicted reformulation categories for $Q_t$</td>
</tr>
<tr>
<td>$s \in S$</td>
<td>RCS states</td>
</tr>
<tr>
<td>$a \in A$</td>
<td>RCS actions</td>
</tr>
<tr>
<td>$R^t_i$</td>
<td>RCS reward for $Q^t_i$</td>
</tr>
<tr>
<td>$\Phi_i(q_1 \ldots q_n)$</td>
<td>Function to map $q_i$ to state space</td>
</tr>
<tr>
<td>$M(s, A)$</td>
<td>Action masking vector</td>
</tr>
<tr>
<td>$Q(s, a)$</td>
<td>Q-value (expected reward) for $a$ in $s$</td>
</tr>
<tr>
<td>$Q^*(s, a)$</td>
<td>Optimal Q-value</td>
</tr>
<tr>
<td>$\pi$</td>
<td>RCS policy</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Discount factor in Q-Learning</td>
</tr>
<tr>
<td>$W_1, W_2$</td>
<td>Weight matrices in RCS Deep Q-Network</td>
</tr>
<tr>
<td>$h$</td>
<td>Hidden vector size</td>
</tr>
<tr>
<td>$d$</td>
<td>Dimensionality of input encoding vector</td>
</tr>
<tr>
<td>$\text{Prob}(\cdot)$</td>
<td>Probability</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Boltzmann temperature for action sampling</td>
</tr>
</tbody>
</table>

**Conversation.** A conversation $C$ consists of a sequence of $(Q_t, A_t)$ turns around a topic of interest. An example is in Sec. 1.

**Intent.** An intent is a specific information need: a conversational question and its reformulations share the same intent. In this work, each training and test question in a benchmark represents a unique intent: the reformulations of the training and test cases have different surface forms while preserving the original intent.

**Question.** A question $Q$ manifests an intent and consists of a sequence of tokens $(q_1 \ldots q_n)$. $Q$ can either be complete (explicit expression of intent), like *What’s the 2022 LOTR TV series called?* ($Q_1$), or incomplete (implicit expression of intent), like *And who in the Jackson trilogy?* ($Q_4$).

**Answer.** An answer $A$ is a response to the information need in question $Q$ (*Harry Sinclair* is the answer $A_4$ to $Q_4$). In this work, an answer can be a KG entity, a type, or a literal. An answer here can be a ConvQA model’s response or a gold answer from a benchmark.

**Reformulation.** A question reformulation is obtained by transforming a question into a different surface form with the same intent. A reformulation is generated using an *(operation, operand)* pair and the original question. Here, operations could be insertion, deletion, substitution, while operands could be entities, predicates, question entity types, expected answer types. An example transformation is adding an answer type to question $Q_2$: *TROP* airing on?; to produce the reformulation $Q_2^1$: *Network* *TROP* airing on?.

**Mention.** A mention refers to a sequence of tokens in $Q$ that is the surface form of a KG item (entity, predicate, or type). A mention of a predicate is referred to as a relation. For example, in $Q_2^1$: *Network* *TROP* airing on?; *Network*, *TROP*, and *airing on* are mentions of KG answer type video streaming service, KG entity The Rings of Power (*TROP*), and KG predicate original broadcaster, respectively.

**Figure 2: Workflow of REIGN: RCS is trained by reinforcement learning, and RG by supervised learning.**

**3 THE REIGN FRAMEWORK**

An overview of the workflow in the proposed REIGN architecture is depicted in Fig. 2. The pipeline consists of three trainable models, where the first two are our contributions:

- A reformulation category selector (RCS) model, that takes a question $Q_t$ as input, and produces a reformulation category $RC^t_i$ for transforming $Q_t$, as output.

- A reformulation generator (RG) model, that takes some $Q_t$ and $RC^t_i$ as input, and produces a reformulation $Q_i^t$ of $Q$ according to $RC^t_i$, as output.

- An external ConvQA model, that takes some $Q_t$ as input, and produces a ranked list of answers $(A_t)$ as output.

Reinforcement learning (RL) is used to train the RCS model (Deep Q-Networks [49] in this work), with the goal of learning to select the most suitable transformation categories given a specific question, using existing QA performance metrics or suitable alternatives as reward signals. The categories come from our novel reformulation taxonomy. The RG model is trained with (distantly) supervised learning (SL), using an LLM (BART in our case [39]) fine-tuned with questions paired with a specific category and the resulting reformulation in the form $(\langle Q_t, RC^t_i \rangle; Q_i^t)$. This is distant supervision in the sense that the reformulations used for fine-tuning are generated in a noisy manner using rules following our taxonomy, and are not human reformulations. The ConvQA model used could be trained with SL [15, 33, 67] or RL [35], according to its original training paradigm. In Fig. 2, the original model ConvQA$_{\text{orig}}$ is trained with $(Q_t, A_t)$ pairs in a ConvQA benchmark, while the more robust model ConvQA$_{\text{robust}}$ is trained on additional QA pairs where the reformulations $(Q_i^t)$ for a specific $Q_t$ are also paired with the original gold answer $A_t$. We now describe each component.

**4 REFORMULATION CATEGORY SELECTOR**

**4.1 Reformulation taxonomy**

**Categories.** We propose a taxonomy of reformulations, a topic that has mostly been treated as monolithic in past work [9, 27, 35]. To begin with, observe that a reformulation of a conversational question is a *modification* of its basic parts. Thus, a systematic generation of reformulations involves an understanding of these parts and meaningful modifications. For (Conv)QA over KGs, these basic question components comprise mentions of one or more entities, their types, predicates, and expected answer types. In analogy with string edit operations, our modifications include insertion, deletion and substitution. Transposition could be another basic operation,
The last operation can be sub-divided into three categories as per reformulation categories: (i) we deal with factoid QA, and there are usually only a few corresponding to valid actions, and zeros elsewhere, is element-wise multiplied with the vector containing learnt probabilities of actions at a given state. Transitions. The transition function \( \delta \) is deterministic and updates the state \( s \in S \) by applying one of the actions \( a \in A \) resulting in a new state \( s' \in \delta \) that corresponds to the encoding of the reformulation \( Q' \). The resulting reformulation is obtained by invoking the RG model (line 6 in Algorithm 1).

Rewards. The reward \( R \) models the quality of the chosen action, and guides the agent towards its goal, which here is an improved answering performance. When a selected category leads to a reformulation on which the ConvQA model obtains better performance than the original question, the agent should get a high reward, and vice versa (line 8 calls the ConvQA model for this reward). Thus, an obvious choice here is to use any desired QA performance metric as the reward. We use the reciprocal rank (RR) \([76]\) metric in this work, that is the reciprocal of the first rank at which a gold answer is found. We use it for the following reasons: (i) we have binary relevance of response entities, either correct (1) or incorrect (0); (ii) we deal with factoid QA, and there are usually only a few correct answers (typically between one and three for the benchmarks used). Formally, this reward based on reciprocal rank difference is computed as:

\[
R = \text{ReciprocalRank}(\langle A^*_q \rangle) - \text{ReciprocalRank}(\langle A \rangle)
\] (1)

where \( \langle A_q \rangle \) and \( \langle A^*_q \rangle \) are the ranked lists of responses by the ConvQA model to \( Q \) and \( Q^*_q \), respectively. If the correct answer was not found in the top few positions (five in our experiments), then we set the reciprocal rank value to \(-1\). The range of this reward then lies in the closed interval \([-2, +2]\). Since this nicely corresponds to a symmetric positive reward and punishment in respective cases of success and failure, we do not perform any further reward normalization. Note, however, that our framework can be used with any metric of choice: the use of RL removes the dependency on the metric being differentiable.

Algorithm. As motivated before, we use Deep Q-Networks (DQN) as our RL algorithm, which is a model-free, value-based method that with an environment (the reformulation generator and the ConvQA model, described later). This environment provides the next state \( s' \in S \) (the generated reformulation \( Q' \)) and a reward \( R \) (QA performance metric) to the agent. This is a Markov Decision Process (MDP) comprising states, actions, the transition function, and rewards \((S, \mathcal{A}, \delta, R)\), where the individual parts are defined next. Algorithm 1 shows the precise steps of applying Deep Q-Learning in the RCS model.

States. A state \( s \in S \) is defined by a conversational question, represented by its encoding with function \( \Phi \): \( s = \Phi(Q) \) (lines 3, 7 in Algorithm 1; BERT \([19]\) embeddings in our experiments, averaged over each question token, and over all hidden layers).

Actions. The set of actions \( \mathcal{A} \) corresponds to the 15 reformulation categories from our taxonomy. Note that every action (category) may not be available at every state. For instance, when a question does not have any mention of an entity type, it is not meaningful to apply the actions of deletion or substitution of an entity type. Therefore, we use action masking as follows to allow only valid actions to be chosen given the current state (when this information is available). A masking vector \( M(s, \mathcal{A}) \) that has ones at indices corresponding to valid actions, and zeros elsewhere, is element-wise multiplied with the vector containing learnt probabilities of actions at a given state.

Rewards. The reward \( R \) models the quality of the chosen action, and guides the agent towards its goal, which here is an improved answering performance. When a selected category leads to a reformulation on which the ConvQA model obtains better performance than the original question, the agent should get a high reward, and vice versa (line 8 calls the ConvQA model for this reward). Thus, an obvious choice here is to use any desired QA performance metric as the reward. We use the reciprocal rank (RR) \([76]\) metric in this work, that is the reciprocal of the first rank at which a gold answer is found. We use it for the following reasons: (i) we have binary relevance of response entities, either correct (1) or incorrect (0); (ii) we deal with factoid QA, and there are usually only a few correct answers (typically between one and three for the benchmarks used). Formally, this reward based on reciprocal rank difference is computed as:

\[
R = \text{ReciprocalRank}(\langle A^*_q \rangle) - \text{ReciprocalRank}(\langle A \rangle)
\] (1)

where \( \langle A_q \rangle \) and \( \langle A^*_q \rangle \) are the ranked lists of responses by the ConvQA model to \( Q \) and \( Q^*_q \), respectively. If the correct answer was not found in the top few positions (five in our experiments), then we set the reciprocal rank value to \(-1\). The range of this reward then lies in the closed interval \([-2, +2]\). Since this nicely corresponds to a symmetric positive reward and punishment in respective cases of success and failure, we do not perform any further reward normalization. Note, however, that our framework can be used with any metric of choice: the use of RL removes the dependency on the metric being differentiable.

Algorithm. As motivated before, we use Deep Q-Networks (DQN) as our RL algorithm, which is a model-free, value-based method that

Figure 3: Taxonomy of reformulation categories. Legend: part = question-part; INS = Insert, DEL = Delete, SUBS = Substitute; ent = entity mention, rel = relation, ent-type = entity type mention, ans-type = answer type mention; w/ = with.

but we do not consider that in this work as reordering question phrases has little effect on several retrieval models. Viewing these three operations and the four parts of a question as operands, we obtain a taxonomy as shown in Fig. 3, where reformulation categories are leaf nodes (marked orange). Examples are "INSERT entity-type", "SUBSTITUTE relation", and "DELETE relation". Note that we require our reformulations to be intent-preserving: this imposes constraints on what we can insert or substitute in the original question. We cannot, for example, replace an entity or relation by a different one that would disturb the semantics of the conversation as a whole.

Phenomena. As shown with dashed boxes in Fig. 3, our taxonomy subsumes several classes of conversational phenomena:

- Insertions complete the question to a more intent-explicit form;
- Deletions cause ellipses in context;
- Substitutions create paraphrases;
- Substituting entity mentions specifically leads to coreferencing.

The last operation can be sub-divided into three categories as per the case of substitution with a pronoun ("TROP" \(\mapsto\) "it") or with its type ("TROP" \(\mapsto\) "the series") or with an alias ("TROP" \(\mapsto\) "Rings of Power"). A special case in our taxonomy is the operation "RE-TAIN whole question", where the question is left as such: it can be considered as a degenerate reformulation. Finally, we have 15 reformulation categories, corresponding to these leaf nodes.

4.2 Training the RCS model

Overall idea. Given an input question and the taxonomy, we would like the Reformulation Category Selector (RCS) model to suggest some categories so that training with reformulations that belong to these categories would lead to better QA performance metrics. This, in turn, means that we would like the RCS to estimate values that correspond as much as possible to such metrics. This motivates us to use Deep Q-Networks (DQN) \([49]\), a reinforcement learning approach that directly learns a value function approximator using QA metrics as rewards. The estimate of the value of a state, action pair is in turn used to infer the policy for predicting or sampling actions. This is in contrast to the relatively more popular choice of learning policy gradients that directly model action probabilities given an input state (for example, the REINFORCE algorithm \([79]\)).

Concretely, we employ DQNs to train an agent (the RCS model) to select actions \( a \in \mathcal{A} \) (reformulation categories) given a current state \( s \in S \) (the input question \( Q_\ell \)). The agent interacts...
learns to predict so-called Q-values \( Q(s, a) \) for each state-action pair to quantify the usefulness of taking action \( a \) in state \( s \) under a policy \( \pi \) [49]. The policy is a function mapping states to actions based on the Q-values. The main update step in Q-Learning is:

\[
Q(s, a) \leftarrow Q(s, a) + \alpha \cdot [R + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)]
\]  
\( \alpha \) is the step size and \( \gamma \) is the discount factor that determines how much influence the next state’s estimate has on the current state. The term \( [\cdot] \) is called TD target. \( Q(s, a) \) is randomly initialized, except for terminal states, where this is zero. In practice, the parameters of the DQN are updated batch-wise (lines 10 – 18). A batch consists of a set of experiences: each experience is a tuple of the form \((s, a, r, s')\) (line 9). The updated parameters \( \theta \) are obtained by calculating the Mean-Squared Error between the TD target and the current Q-values of each state-action pair in the batch (line 19).

The objective function is to maximize the expected reward:

\[
Q^*(s, a) = \max_{a} Q_{\pi}(s, a) = \max_{a} \mathbb{E}_{\pi} [R|s, a]\]

where \( Q^*(s, a) \) is the optimal Q-value for a specific \((s, a)\) pair. Since our state space is large, we cannot directly learn tabular entries for each \((s, a)\) pair, as was typical in more traditional RL setups. Instead, Q-values are predicted via a neural Q-network with trainable parameters \( \theta \) (a two-layer feed-forward network in our case):

\[
\begin{align*}
Q^2(s, a) &= \max_{a} Q_\pi(s, a) = \max_{a} \mathbb{E}_{\pi} [R|s, a] \\
Q_\pi(s, a) &= \max_{a} \mathbb{E}_{\pi} [R|s, a]
\end{align*}
\]

where \( Q_\pi(s, a) \) is a function returning a vector of size \( |\mathcal{A}| \) and stores the obtained values for every action \( a \in \mathcal{A} \) given some \( s \in \mathcal{S} \), \( s \in \mathbb{R}^{d \times 1} \); \( d \) is the size of the input encoding vector; \( W_1 \in \mathbb{R}^{d \times h} \), \( W_2 \in \mathbb{R}^{h \times |\mathcal{A}|} \) are the weight matrices; \( M(s, \mathcal{A}) \in \mathbb{R}^{1 \times |\mathcal{A}|} \) is the action mask; and hidden size \( h \) is a tunable hyperparameter. ReLU is the non-linear activation function.

During training, the agent needs to explore different actions in each state via sampling. In this work, we sample from a Boltzmann distribution to enable such exploration (line 5). A Boltzmann distribution is parameterized by a temperature \( \tau \) that we can use to conveniently control the degree of exploration:

\[
\text{Prob}(a^{\text{sample}}_i, r) = \frac{e^{Q_\pi(s, a^{\text{sample}}_i)} / r}{\sum_{a \in \mathcal{A}} e^{Q_\pi(s, a) / r}}
\]

A \( r \)-value close to zero means taking the best action (with highest reward at this point) greedily more often, whereas larger values (\( r \) is unbounded) make the actual Q-values less relevant and result in a random policy.

### 4.3 Applying the RCS model

We train the RCS on the development set\(^1\) of a ConvQA benchmark, and apply it on the questions in the training set. At RCS inference time, the agent follows a greedy policy \( \pi \) with respect to Q-values, and typically chooses an action \( a^{\text{greedy}} \) in a state \( s \) as below:

\[
a^{\text{greedy}} = \pi_s = \arg \max_{a \in \mathcal{A}} Q(s, a)
\]

\( \pi \) is the action mask; and hidden size \( h \) is a tunable hyperparameter. ReLU is the non-linear activation function.

\( ^1 \)In this work, we reuse ConvQA benchmark development sets for training the RCS model, fine-tuning the RG model, adjusting hyperparameters for all models, and selecting best Razin configurations. We intentionally avoid large-scale learning of RCS and RG on train sets to stress-test generalizability of Razin components: only the QA model is learnt from the full train set. Further, any kind of leakage to test sets is thereby precluded for all models.

#### 5 REFORMULATION GENERATOR

##### 5.1 Training the RG model

**Basic setup.** The reformulation generator (RG) is implemented by fine-tuning a pre-trained LLM for sequence generation (BART [39] in our case). BART is especially effective when information is both copied from the input plus perturbed with noise, to generate the output autoregressively [39]: this is exactly the setup in this work. The concatenation of the conversation history \( Q_t \) and the current question \( q_t \), and a special reformulation category tag ("rc1", "rc2", ... "rc15") constitute the input, and the category-specific reformulation is the output.

**Noisy data for fine-tuning.** We generate fine-tuning data for the BART model through *distant supervision*. This is a noisy process, but the alternative of strong supervision would entail the use of human-generated reformulations. This would be expensive to obtain at scale (benchmarks like ConvRef [35] contain a relatively small number of unique reformulations, and lack category labels). Further, given a conversational question, an average crowdworker is not likely to be able to come up with several diverse and distinct reformulations for each category. Specifically, we adopt the following strategy. Given a question, we run the recently proposed Clocq

---

**Algorithm 1: Deep Q-Learning in RCS model**

**Input:** Sequence of conversational questions \( Q \), step size \( \alpha > 0 \), discount factor \( \gamma > 0 \), Boltzmann temperature \( \tau > 0 \), size of update \( \text{batchSize} \), initial DQN parameters \( \theta \)

**Output:** Updated DQN parameters \( \theta \)

1. \( \text{experience} \leftarrow () \) \( \triangleright \) Initialize experience queue
2. foreach \( Q_t \in Q \) do
   3. \( s \leftarrow \Phi(Q_t) \) \( \triangleright \) Encode question
   4. \( Q_\theta(s, a) = M(s, \mathcal{A}) \odot (W_2 \times \text{ReLU}(W_1 \times s)) \) \( \triangleright \) Get Q-values for actions in \( s \)
   5. \( a^{\text{sample}} \leftarrow \text{Prob}(A, r) \) \( \triangleright \) Use Eq. 5
   6. \( Q_\theta^2(s, a) \leftarrow \text{RG}((Q_1, \ldots, Q_t, R^{a^{\text{sample}}}) \) \( \triangleright \) Invoke RG
   7. \( s' = \Psi(Q_\theta^2) \) \( \triangleright \) Encode reformulation
   8. \( r \leftarrow (\text{ConvQA}(Q_t)) \) \( \triangleright \) Invoke ConvQA to obtain reward
   9. \( \text{experience}.enqueue(s, a^{\text{sample}}, s', r) \) \( \triangleright \) Store experience
10. if \( \text{experience} \gg \text{batchSize} \) then
      11. \( \text{batch} \leftarrow \text{experience}.dequeue(\text{batchSize}) \)
      12. \( q^{\text{batch}} \leftarrow () \) \( \triangleright \) Initialize queue for storing Q-values
      13. \( q^{\text{targets}} \leftarrow () \) \( \triangleright \) Initialize queue for storing TD targets
      14. foreach \( (s, a, s', r) \in \text{batch} \) do
          15. \( q^{\text{batch}}.enqueue(q, a^{\text{sample}}) \)
          16. \( q^{\text{targets}}.enqueue(r + \gamma \cdot \max_{a'} Q_\theta(s', a')) \)
      17. end
      18. \( \theta \leftarrow \theta + \alpha \nabla (\text{batchSize} Q^{\text{targets}} - q^{\text{batch}})^2 \)
19. return \( \theta \)

In our case, the RCS predicts the top- \( k \) reformulation categories \( \{RC^i_j\} \) for each training question \( Q_t \), which the RG picks up to actually generate the new question variants \( \{Q^i_j\} \).

---
API [13] to obtain mentions and disambiguations of entities and predicates. Types are also essentially entities and hence detected in this step: one can detect whether an entity is a KG type by searching the KG for a fact where this entity is an object and the predicate indicates a type relationship (instance of in Wikidata). We look up KG types of disambiguated question and answer entities to decide if the detected type mentions correspond to question entity types or expected answer types.

Once we have mentions and their disambiguations, we can apply our transformations from the taxonomy (Fig. 3) on input questions. Deletion is straightforward: the mention is simply removed the question token sequence. For substitution, the main decision to make is the source of alternative surface forms for the linked KG items. We use aliases in curated KGs for synonyms of entities and predicates, a rich and precise yet relatively under-explored source. Substitution happens in-place: the source mention is replaced by the target mention from the KG alias list in its corresponding position in the question. Each unique alias results in a unique transformation possibility. Pronoun replacements for human entities are performed by looking up their gender in the KG. For insertions, the main concern is the position of insertion in the question: (i) mentions of answer types and relations are inserted just after the question word; (ii) mentions of entity types are inserted just before the respective entity; and (iii) entity mentions are inserted at the end of the question. Exact details are in our source code repository.

5.2 Applying the RG model

The BART model is fine-tuned on distantly supervised data generated with the ConvQA dev set. It is then applied on the train set where a question and a category from the RCS are already available.

6 CONVERSATIONAL QUESTION ANSWERING

6.1 Training the ConvQA model

A ConvQA model is trained on sequences of \( \langle Q_t, A_t \rangle \) pairs. In the original training mode, QA pairs are directly used from the benchmark train sets. This original or initial QA model is used to collect rewards for the RCS in one pass over the dev set (as mentioned earlier, the QA dev set is used to train the RCS model). After the trained RCS and RG models generate the reformulations for each training question, these reformulations are paired with the corresponding gold answer of the original training question. These new (reformulation, gold answer) pairs are added to the benchmark, and the ConvQA model is trained again on this augmented resource. This model is expected to be more robust than the original model (original and robust models are marked ConvQA\textsubscript{orig} and ConvQA\textsubscript{Robust} in Fig. 2, respectively).

6.2 Applying the ConvQA model

The trained ConvQA model is directly applied to the questions in test sets at answering time to produce ranked lists of entities.

7 EXPERIMENTAL SETUP

Benchmarks. As shown in Table 2, we use two ConvQA benchmarks: CONVMIX [14] (more recent) and CONVQUESTIONS [12] (more popular). These contain realistic questions from crowdworkers. We obtained 20 reformulations from ChatGPT (gpt-3.5-turbo model) for each test question in these benchmarks, with ten each from two different settings: (i) one asked for reformulations with access to the full conversation history (previous questions and gold answers), and (ii) the other only with the current question. Examples are in Table 3. All GPT-generated reformulations are available at our website https://reign.mpi-inf.mpg.de. For prompting GPT, we tried a few alternatives. We saw that examples did not have a noticeable effect on the generations. Thus, we used the following zero-shot prompt (the ‘History’ line is omitted for generating the variants without history) and set the temperature value to zero, for obtaining deterministic behavior (as far as possible):

```
Reformulate the ‘Question’ 10 times in a short, informal way. Assume third person singular if not obvious from the question.

‘History’: [CONVERSATION HISTORY]
‘Question’: [QUESTION]
‘Reformulation’:
```

The second sentence was used to avoid generations like Your place of birth? instead of the correct His/her...? or Her...? There are no duplicates in any of the ChatGPT reformulations. Conversations in CONVQUESTIONS are generated by permuting questions from a seed set of 700 conversations: we used only the train set for this seed (420 conversations) for training ConvQA models, to decouple the effect of data augmentation inherent in the benchmark.

Baselines. ConvQA models belong to two families, one based on history modeling, and the other on question completion (Sec. 9). We choose one open-source system from each family for KG-QA: Conquer [35] (history modeling with context entities, with RL) and the very recent EXPLAIGN (completion to an intent-explicit structured representation, with GNN). EXPLAIGN was built for heterogeneous sources, and we use the KG-only model, in line with our setting. Default configurations were used for both systems.

Metrics. All methods produce ranked lists of entities with binary relevance. We thus used three appropriate KG-QA metrics [64]: Precision@1 (P@1), Mean Reciprocal Rank (MRR), and whether a correct answer is in the top-5 (Hit@5). We define a new metric Robust, that computes, for each question, the number of reformulations (out of 20 here) correctly answerable by a ConvQA model, averaged over the number of test intents. The higher this value, the more robust the model. Statistical significance (*) is conducted via McNemar’s test for binary variables (P@1 and Hit@5), and 2-tailed paired t-test otherwise (MRR, Robust), with \( p < 0.05 \).

Initializing Reign. We use Wikidata as our KG: all models use the dump from 31–01–2022. We use BART (bit.ly/3N9WPvJ, for RG), and BERT (bit.ly/3NkRsd, for state encoding in RCS) implementations from Hugging Face. As history input to BART, we used only the first and previous turns of the conversation [59, 75]. Hyperparameters for the Deep Q-Network in the RCS were tuned on the ConvMix dev set: \( d = 768 \), hidden size \( h = 128 \), Boltzmann temperature \( \tau = 0.3 \), discount factor \( \gamma = 1.0 \) (no decay for future
8 RESULTS AND INSIGHTS

8.1 Key findings

Reign results in robust training. The four methods Conquer, Conquer + Reign (Conquer coupled with Reign), Explaignn, and Explaignn + Reign (Explaignn coupled with Reign) are evaluated on the two benchmarks ConvMix and ConvQuestions. Results on test sets are in Table 5. A clear observation is that methods interfaced with Reign systematically outperform the original ConvQA models, on all test sets and metrics. While numbers are reported on the original test sets for completeness, results become much more significant on the GPT-test sets, with p-values of the order of $10^{-80}$ (recall that these values are averaged over $\approx 100k$-$200k$ cases, Table 2). Importantly, versions with Reign score systematically higher on the robustness metric (Sec. 7), showing that the improved models are capable of handling more lexical and syntactic variations on average (differences higher for larger GPT-sets). ConvQA with these benchmarks and GPT reformulations are challenging: these values are far less than 21 (the Robust measure here lies between 0 and the number of reformulations per question including the original reformulation categories, used for training).

Table 3: Examples of GPT reformulations for test sets.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Question: Which was the first book he wrote?</td>
<td>Question: Author of the fiction?</td>
</tr>
<tr>
<td>Ref 1: What book earned John Updike his first Pulitzer Prize?</td>
<td>Ref 1: Creator of the fiction?</td>
</tr>
<tr>
<td>Ref 2: What was the author's first book to win a Pulitzer?</td>
<td>Ref 2: Which individual is author of the fiction?</td>
</tr>
<tr>
<td>Ref 3: Title of John Updike's first Pulitzer Prize-winning book?</td>
<td>Ref 3: Author of the fiction. The Underground Railroad?</td>
</tr>
</tbody>
</table>

Table 4: Examples of Reign-generated reformulations along with respective reformulation categories, used for training.

<table>
<thead>
<tr>
<th>Movies History: Who was the director of The Lord of the Rings? Peter Jackson.</th>
<th>Movies History: Who is the director of The Lord of the Rings? Peter Jackson.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question: Who was the director of the film?</td>
<td>Question: Who directed the film?</td>
</tr>
<tr>
<td>Ref 1: Who directed The Lord of the Rings?</td>
<td>Ref 1: Who directed The Lord of the Rings?</td>
</tr>
<tr>
<td>Ref 2: Who portrayed Frodo Baggins?</td>
<td>Ref 2: Who portrayed Frodo Baggins?</td>
</tr>
<tr>
<td>Ref 3: Who played Frodo Baggins in it?</td>
<td>Ref 3: Who played Frodo Baggins in it?</td>
</tr>
</tbody>
</table>

Table 5: Examples of test sets.

<table>
<thead>
<tr>
<th>Basketball History: Who is the mastermind?</th>
<th>Basketball History: Who was the mastermind?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question: Who is the mastermind of the series?</td>
<td>Question: Who was the mastermind of the series?</td>
</tr>
<tr>
<td>Ref 1: Who is the mastermind of the series?</td>
<td>Ref 1: Who was the mastermind of the series?</td>
</tr>
<tr>
<td>Ref 2: Who is responsible for the series?</td>
<td>Ref 2: Who is responsible for the series?</td>
</tr>
<tr>
<td>Ref 3: Who is the creator of the series?</td>
<td>Ref 3: Who is the creator of the series?</td>
</tr>
</tbody>
</table>

Table 6: Examples of test sets.

<table>
<thead>
<tr>
<th>Books History: Which is John Updike's first novel?</th>
<th>Books History: Which is John Updike's first novel?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question: What is John Updike's first novel?</td>
<td>Question: What is John Updike's first novel?</td>
</tr>
<tr>
<td>Ref 1: Who is the creator of the novel?</td>
<td>Ref 1: Who is the creator of the novel?</td>
</tr>
<tr>
<td>Ref 2: What is the date of its publication?</td>
<td>Ref 2: What is the date of its publication?</td>
</tr>
</tbody>
</table>

Table 7: Examples of test sets.

<table>
<thead>
<tr>
<th>TV series History: Who is the director of the series? Jason Bateman.</th>
<th>TV series History: Who is the director of The Lord of the Rings? Peter Jackson.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question: Who is the director of the series?</td>
<td>Question: Who is the director of the film?</td>
</tr>
<tr>
<td>Ref 1: Who is the director of the series?</td>
<td>Ref 1: Who is the director of the film?</td>
</tr>
<tr>
<td>Ref 2: Who is the producer of the series?</td>
<td>Ref 2: Who is the producer of the film?</td>
</tr>
<tr>
<td>Ref 3: Who is the creator of the series?</td>
<td>Ref 3: Who is the creator of the film?</td>
</tr>
</tbody>
</table>

Table 8: Examples of test sets.

<table>
<thead>
<tr>
<th>Music History: Who was the lead singer of the Beatles? John Lennon.</th>
<th>Music History: Who was the lead singer of the Beatles? John Lennon.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question: Who was the lead singer of the Beatles?</td>
<td>Question: Who was the lead singer of the Beatles?</td>
</tr>
<tr>
<td>Ref 1: Who is the lead singer of the Beatles?</td>
<td>Ref 1: Who is the lead singer of the Beatles?</td>
</tr>
<tr>
<td>Ref 2: Who is the guitarist of the Beatles?</td>
<td>Ref 2: Who is the guitarist of the Beatles?</td>
</tr>
<tr>
<td>Ref 3: Who is the drummer of the Beatles?</td>
<td>Ref 3: Who is the drummer of the Beatles?</td>
</tr>
</tbody>
</table>
Reign components are generalizable. Results on the ConvQuestions benchmark showcase successful zero-shot application of Reign modules. Given that the ConvQuestions test sets are much larger than ConvMix (see Table 2), improved results over the original QA modules show that our RCS and RG modules, individually, are immune to idiosyncrasies in specific datasets.

Benefits of Reign hold over domains and turns. We report drilldown results over five domains and individual conversation turns in Tables 6 and 7. We show that the benefits provided by reinforced reformulation generation are not limited to specific domains, or shallow conversation turns only.

### 8.2 In-depth analysis

In Table 8, we report in-depth analyses of the moving parts in Reign, using Conquer on the ConvMix-GPT-set. Trends with Explaignn and ConvQuestions are similar. We do not use this table for making design choices – rather, we expose large-scale effects of sub-optimal configurations: hence the choice of a $\approx 100k$-GPT set instead of the typically small dev set.

---

**Table 7: Turn-wise P@1 results on GPT-ConvMix testset.**

<table>
<thead>
<tr>
<th>Row</th>
<th>Configuration</th>
<th>P@1</th>
<th>MRR</th>
<th>Hit@5</th>
<th>#Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RCS (DOM, top-5) + RG (BART) [Full]</td>
<td>0.190</td>
<td>0.236</td>
<td>0.289</td>
<td>43.6k</td>
</tr>
<tr>
<td>2</td>
<td>RCS (DOM, top-3) + RG (BART)</td>
<td>0.184</td>
<td>0.231</td>
<td>0.288</td>
<td>30.5k</td>
</tr>
<tr>
<td>3</td>
<td>RCS (DOM, top-1) + RG (BART)</td>
<td>0.178</td>
<td>0.228</td>
<td>0.284</td>
<td>15.9k</td>
</tr>
<tr>
<td>4</td>
<td>No RCS (All cats) + RG (BART)</td>
<td>0.188</td>
<td>0.234</td>
<td>0.292</td>
<td>126k</td>
</tr>
<tr>
<td>5</td>
<td>No RCS (Random cats) + RG (BART)</td>
<td>0.182</td>
<td>0.232</td>
<td>0.293</td>
<td>42k</td>
</tr>
<tr>
<td>6</td>
<td>No RCS (Sample cats) + RG (BART)</td>
<td>0.185</td>
<td>0.231</td>
<td>0.287</td>
<td>41.9k</td>
</tr>
<tr>
<td>7</td>
<td>No RCS (INS part) + RG (BART)</td>
<td>0.183</td>
<td>0.230</td>
<td>0.288</td>
<td>42k</td>
</tr>
<tr>
<td>8</td>
<td>No RCS (DEL part) + RG (BART)</td>
<td>0.172</td>
<td>0.218</td>
<td>0.273</td>
<td>42k</td>
</tr>
<tr>
<td>9</td>
<td>No RCS (SUBS part) + RG (BART)</td>
<td>0.183</td>
<td>0.228</td>
<td>0.282</td>
<td>58.8k</td>
</tr>
<tr>
<td>10</td>
<td>No RCS + No RG (Question completion)</td>
<td>0.175</td>
<td>0.224</td>
<td>0.284</td>
<td>15.1k</td>
</tr>
<tr>
<td>11</td>
<td>No RCS + No RG (Question rewriting)</td>
<td>0.180</td>
<td>0.230</td>
<td>0.291</td>
<td>15.1k</td>
</tr>
</tbody>
</table>

**Figure 4: Common category predictions by the RCS DQN.**

RCS with DQN is vital. First and foremost, we show that selecting reformulations with our DQN is necessary, and simply taking all noisy reformulations does not serve as a sledgehammer for performance improvement even at three times the number of data points used (Row 1 vs. Row 4). This makes a solid case for judicious augmentation. Using all reformulations does lead to higher answer recall as seen through the Hit@5 value, but at the cost of precise ranking. Using top-5 reformulations is a sweet spot for deploying the RCS (Row 1 vs. 2 and 3). Using higher numbers drastically increases the training time and often produces degenerate reformulations. Contrast against a random choice of categories inside the RCS is a natural experiment, and we find this to be sub-optimal (see P@1 in Row 5). Another stronger baseline is to sample $k = 5$ categories according to the Q-value distribution: this again falls short of a top-k prediction (Row 6).

The whole taxonomy matters. It may appear that using only insertion or substitution operations from the taxonomy may suffice for robust learning, but we find that considering all categories jointly (Row 1) is superior to using only individual “meta”-categories (INS, DEL SUBS in Rows 7–9). While using only deletion operations hurts performance the most (Row 8), it is thus clear that carefully removing parts of questions also contributes to a stronger model (for example, deleting an entity was considered to improve MRR 10% of the time on ConvQuestions, presumably removing noise). Fig. 4 shows the union of the top-5 frequent predictions from our RCS DQN for the two benchmarks. Insertion of question entity types and expected answer types are generally useful for disambiguation, and substituting relations with aliases naturally makes the system more robust to predicate paraphrasing. The original question was retained 10 – 20% of the time.

**Question rewriting is not enough.** As discussed in Sec. 1, reformulating a conversational question into a more complete form at answering time is a prevalent approach in ConvQA. As such,
comparison with such rewriting or completion approaches is out of scope, as we focus on more robust training. Nevertheless, we explore the natural possibility of using completed forms of questions during training, as opposed to a set of noisy reformulations. The ConvMix benchmark [14] contains intent-explicit questions by the original crowdworkers who generated the conversations, and can thus be treated as gold standard completions. We found that this falls short of our proposed method (Row 1 vs. Row 10), as does question rewriting using T5 [42] (Row 1 vs. Row 11). Interestingly, corroborating findings from the BART experiments, noisy rewrites with T5 outperform human completions. Note that completion or rewriting entails one longer version of the question (hence \( \approx 15k \) data points): we find that generating a small set of potentially incomplete variants with Reign improves performance.

**Intrinsic rewards also work well.** Our DQN uses differences in reciprocal ranks, computed from gold answers in benchmarks, as extrinsic rewards. A natural question is what happens in cases where such relevance assessments are not available. We thus explored an alternative of an intrinsic reward [10, 45] computed as the difference in the ConvQA model’s probabilities of its top-1 answer for the reformulation and the original question [10, 45] (this is analogous to blind relevance feedback). On a positive note, this resulted in comparable performance on the ConvMix dev set (0.270 P@1 for intrinsic vs. 0.269 for intrinsic; 0.311 MRR for both).

**Manual error analysis.** The authors analyzed 10 reformulations from each category for both BART reformulations and the original fine-tuning data (15 × 10 × 2 = 300 in all), to look for potential issues. There were only minor problems detected for both scenarios. The concerns with BART were as follows: unintelligible intent (4 cases), hallucinations (5), wrong category applied (13), information removed unintentionally (15), transformation possible but not made (13), unsuitable entity or type added (4), and information already in the question was added again (5). The concerns with the initial noisy data can sometimes be traced back to incorrect processing of the benchmarks (Sec. 5.1), like wrong predicate (4 cases) and addition of information already present due to incorrect markup (4). Other errors include: intent changed (6), unsuitable types inserted (7 cases), and changes possible but not made (2).

**GPT cannot replace the Reign pipeline.** It is a common trend nowadays to use LLMs like GPT at multiple points in pipelines. We thus checked whether the same ChatGPT model that generated our test set could actually replace the whole Reign pipeline by directly generating reformulations for training questions. Importantly, we found that this underperforms Reign when five reformulations are considered for each alternative, on the original ConvMix dev set (evaluation of GPT reformulations on GPT test sets could result in undesirable biases): 0.270 P@1 for Reign vs. 0.261 for GPT (Conquer), and 0.423 for Reign vs. 0.405 for GPT (ExplainN). Note that the GPT reformulations are model-agnostic: this shows that reformulations generated with model-aware performance feedback is indeed a better choice for robust training.

**9 RELATED WORK**

**Conversational question answering.** ConvQA [11, 12, 63, 64, 66] can be viewed as a research direction under the umbrella of conversational search [16, 53, 84], with natural-language utterances as input. Answers are crisp entities [25, 33], sentences [5], or passages [17, 58]. Methods proposed belong to two major families: they either (i) derive a self-contained version of the question that can be handled by a standard QA system (referred to as rewriting [10, 29, 36, 62, 75, 83]), resolution [37, 77], or even reformulation [50, 74], in different works), or (ii) model the history as additional context to answer the current question [24, 26, 35, 57, 59, 60, 70]. Reign is not a QA model by itself, but can improve the performance of any given ConvQA system: we demonstrate this by choosing one method from each family of approaches in our experiments [15, 35]. In this work, we enhance conversational QA over KGs [25, 31–33, 54, 67], where answers are small sets of entities.

**Robustness in QA.** Improving the robustness or generalizability of ConvQA models has not seen much dedicated activity: work has mostly been limited to specific benchmarks of choice [12, 14, 25, 66, 67]. Implicitly, authors have tried to prove robust behavior by the use of multiple benchmarks [33, 38, 48], or zero-shot application of models to new benchmarks [15]. Data augmentation, given one or more reformulations, is one of the prominent approaches for increasing model robustness in QA [5, 6, 44, 56, 65, 68, 81]. Our work stands out as model-specific data augmentation, a philosophy for effective training by trying to fill “gaps” in a specific model’s learned behavior, instead of feeding in a very large volume of noisy data to all models. Some recent works in QA over text investigate model robustness by perturbing input passages [24, 51], while we tap into question reformulations as a perturbation on the question-side.

**Reformulations in search and QA.** Work on question or query reformulations in search [40, 52, 55, 80], QA [9, 27, 35, 47, 71], and recommenders [85], can be broadly positioned in a 2 × 2 space according to the definition of a reformulation:

- Rephrasing (of the same intent) [20, 21, 35, 72] versus refinement (into a variation of the previous intent) [46, 47, 52, 85].
- Using for better training [40, 55] versus using for better inference [9, 18, 52, 80].

This work falls into the rephrasing-for-training quadrant, viewing reformulations as rephrased user utterances for the current question in a conversation, and leveraging these for training a more robust model. Early work on automatic acquisition of query reformulation patterns [47, 71, 72, 80], or on paraphrasing for improving model robustness [1, 2, 8, 20–23], did not account for answers from previous turns, and more generally, did not address the specific difficulty of incomplete and ad-hoc user utterances in conversations.

**10 CONCLUSION**

This work contributes a method that makes conversational question answering models more robust with generated reformulations that are specifically guided towards better QA performance. The proposed framework judiciously picks the most suitable choices for enhanced training, as opposed to brute-force data augmentation with all possible reformulations. Experiments with two state-of-the-art ConvQA methods demonstrate benefits of the Reign method.

**ACKNOWLEDGMENTS**

We thank Philipp Christmann from the Max Planck Institute for Informatics for useful inputs at various stages of this work.
ETHICAL CONSIDERATIONS

There are no negative ethical and societal concerns arising from this work. The questions and reformulations are based on public benchmarks, and do not contain any sensitive or personally identifiable information (PII). Prompts for ChatGPT are not meant to evoke adversarial, hateful, or malicious responses. Security, safety, and fairness concerns are not applicable as well.

REFERENCES


