Abstract

Knowledge-grounded dialogue systems utilise external knowledge such as knowledge graphs to generate informative and appropriate responses. A crucial challenge of such systems is to select facts from a knowledge graph pertinent to the dialogue context for response generation. This fact selection can be formulated as path traversal over a knowledge graph conditioned on the dialogue context. Such paths can originate from facts mentioned in the dialogue history and terminate at the facts to be mentioned in the response. These walks, in turn, provide an explanation of the flow of the conversation. This work proposes KG-CRUSE, a simple, yet effective LSTM-based decoder that utilises the semantic information in the dialogue history and the knowledge graph elements to generate such paths for effective conversation explanation. Extensive evaluations showed that our model outperforms the state-of-the-art models on the OpenDialKG dataset on multiple metrics.

1 Introduction

Inducing factual information during response generation has garnered a lot of attention in dialogue systems research. While language models (Zhao et al., 2020; Zheng et al., 2020) have been shown to generate responses akin to the dialogue history, they seldom contain factual information, leading to a bland conversation with the agent. Knowledge-grounded dialogue systems focus on leveraging external knowledge to generate coherent responses. Knowledge Graphs (KGs) are a rich source of factual information and can be combined with an utterance generator for a natural and informative conversational flow.

Zhou et al. (2018) showed that utilising KGs in dialogue systems improves the appropriateness and informativeness of the conversation. Augmenting utterances in a dialogue with the KG information guides the conversational agent to include relevant entities and facts in the response. For example, Figure 1 shows an example conversation where a user is interacting with a dialogue agent about movies. The agent has access to a KG that aids in suggesting relevant facts during the dialogue flow. When responding to utterance 3, the agent can utilise information from the KG and produce relevant facts about “Christopher Nolan”. This information would be more engaging than responding with information about “Batman” or “Batman Begins”.

While KGs have been used extensively to include relevant facts in a dialogue, the explicability of such systems is limited. Naturally, this fostered research on developing models for explainable conversation reasoning. Moon et al. (2019) addressed this problem by inducing KG paths for conversation explainability. They posited a dialogue-KG path aligned corpus wherein utterances are augmented with a KG path to denote fact transitions in the dialogue. The KG paths emanate from entities

Figure 1: An example conversation wherein the agent utilises relevant information from the KG while generating responses. The agent generates facts about “Christopher Nolan” in utterance 4 while utilising the semantic information in the dialogue history and the KG.
or facts mentioned in the dialogue history and terminate at the entity to be mentioned in the response text. Such paths form a sequence of entities and relations and aid the dialogue agent in introducing appropriate knowledge in the dialogue. In addition to this, they proposed an attention-based recurrent decoder over the KG to generate entity paths. Jung et al. (2020) designed a novel dialogue-context infused graph neural network to propagate attention scores over the knowledge graph entities for KG path generation. While such approaches have their inherent strengths, their limitations are manifold.

Given a dialogue context, it is desirable to generate paths that result in a natural dialogue flow. Therefore it is essential to capture the semantic information in the dialogue context as well as the KG elements. Transformer based models (Devlin et al., 2019; Lan et al., 2020; Liu et al., 2019a) have enabled the capture of contextual relationships between different words in a sentence. Textual representations from such models have been successfully adapted for the dialogue conditioned KG reasoning task (Jung et al., 2020). However, prior works use the embedding of the [CLS] token to encode the dialogue history and the KG elements. Reimers and Gurevych (2019) demonstrated that such sentence embedding are sub-optimal and lead to degraded performance in downstream applications. Sentence-transformers (Reimers and Gurevych, 2019) are strong tools for capturing the semantic information of a sentence into a fixed-size vector. As KG elements can be long phrases, KG-C RuSE utilises the rich sequential information in the dialogue history and the KG elements for capturing their semantic information.

As a result of the long tailed distribution of node neighbors in a KG, it can become difficult to generate relevant paths over the KG for explainable conversation. Given the dialogue history, it is desirable to traverse paths that are semantically relevant. KG-C RuSE utilises the rich sequential information in the dialogue history and the path history to sample the top-k semantically similar neighbors for extending its walk over the KG.

We show that our KG-C RuSE improves upon the current state-of-the-art on multiple metrics, demonstrating the effectiveness of KG-C RuSE for explainable conversation reasoning.

To summarise, our contributions are as follows:

- We propose a KG-C RuSE, a LSTM based decoder leveraging Sentence-Transformer (SBERT) embedding to reason KG paths for explainable conversation.
- We show the efficacy of our model by improving the current state-of-the-art performances over multiple metrics on the OpenDialKG (Moon et al., 2019) dataset. Additionally, we conduct extensive empirical analysis to emphasise the effectiveness of KG-C RuSE for the reasoning task.
- We release our system and baseline systems as an open-source toolkit to allow reproducibility and future comparison on this task.

## 2 Related Work

The use of external knowledge in dialogue agents has become commonplace, owing to the rich heterogeneous information contained in them. He et al. (2017) addressed the knowledge-grounded conversation task by iteratively updating the knowledge base embeddings to generate informative responses. Following this, knowledge-based dialogue systems have been studied extensively including the collection of new knowledge-grounded datasets (Ghazvininejad et al., 2018; Qin et al., 2019; Zhang et al., 2018) and developing knowledge-centric dialogue systems (Liu et al., 2018; Parthasarathi and Pineau, 2018a; Zhang et al., 2020).

Young et al. (2018a) attempted to integrate a large scale KG into an end-to-end dialogue system. Other similar works (Chen et al., 2019; Zhou et al., 2020; Sarkar et al., 2020) leveraged graph neural networks and KG embeddings to recommend relevant products in conversational recommender systems. Though successful in retrieving suitable entities or facts from the KG, these systems fail to provide explainability to the recommendations.

Such limitations encouraged explainable conversation reasoning using external knowledge. Liu et al. (2019b) develop the problem as a Partially Observable Markov Decision Process and use policy gradient for training the agent to generate KG paths. Moon et al. (2019) posited a KG path-parallel-dialogue corpus along with DialKG Walker (DKGW) model, a recurrent decoder model to generate the KG path for a response entity selection. Jung et al. (2020) suggested the use of graph neural networks using attention flow to generate KG entity paths. While novel, DKGW does not explicitly utilise the graph structure during model training. On the other hand, the performance of AttnIO
(Jung et al., 2020) relies on the node sampler during training. AttnIO becomes computationally expensive due to dialogue specific graph neural network (both during training and inference). To counter these issues, we design a very simple, lightweight, yet efficient LSTM network leveraging the dialogue and path history to extend the path over the KG.

While, DKGW uses TransE (Bordes et al., 2013) for encoding the elements of the KG, such translation embeddings have weak representation capacity. On the other hand, Jung et al. (2020) utilise the ALBERT (Lan et al., 2020) representation of sentence to encode the dialogue history and the KG elements. They use the [CLS] token representation of the text sequence as the sentence representation. Reimers and Gurevych (2019) suggested Sentence-Transformers for encoding sentences. We encode the dialogue history and the KG elements using Sentence-Transformers to capture rich semantic similarities between the dialogue history and the KG elements.

The processing of semantically rich sequential information using a lightweight LSTM model makes KG-CRuSE ideal for generating walks over a KG for explainable conversation.

3 Methodology

In the following sections, we begin with formally introducing the problem statement. We then outline the embeddings used in KG-CRuSE. Following this, we discuss the architecture of KG-CRuSE as illustrated in Figure 2. Finally, we describe decoding process used by KG-CRuSE during the inference step.

3.1 Formal Problem Definition

We describe the problem statement similar to Moon et al. (2019). The KG is defined as $G = \mathcal{V}_K \times \mathcal{R}_K \times \mathcal{V}_K$, where $\mathcal{V}_K$ is set of entities and $\mathcal{R}_K$ is set of relations in the KG. Facts in the KG are denoted by triples, and each has the form $(e, r, e')$ where $e, e' \in \mathcal{V}_K$ are entities and $r \in \mathcal{R}_K$ is the relation connecting them.

In addition to the KG, each input contains a dialogue $D \in \mathcal{D}$, represented as a sequence of utterances $D = \{s_1, ..., s_n\}$, and the set of entities $x_e = \{x^{(i)}_e\}$ occurring in the user’s last utterance $s_n$, where $x^{(i)}_e \in \mathcal{V}_K$. The output is represented as $y = \{y_e, y_r\}$, where $y_e$ is a set of entity paths $y_e = \{y^{(i)}_e\}$, with each element $y^{(i)}_e = \{y^{(i)}_{e,t}\}_{t=1}^T$ denoting an entity path connecting $x^{(i)}_e$ to the response entity $y^{(i)}_{e,r}$. Likewise, $y_r = \{y^{(i)}_r\}$ is a set of relation paths, where $y^{(i)}_r \in \mathcal{R}_K$. The element $y^{(i)}_r = \{y^{(i)}_{r,t}\}_{t=1}^T$ is a sequence of relations from the KG connecting $x^{(i)}_e$ and $\{x^{(i)}_{e,t}\}_{t=1}^T$.

3.2 Dialogue and KG Representation

Capturing the semantic information in the dialogue context is an important component of our model. SBERT is a contextual sentence encoder that captures the semantic information of a sentence in a
fixed-size vector. We encode pieces of text using Equation 1. The text is first sent through a pre-trained BERT model to obtain the contextual representation of its tokens. The sentence embedding is computed by taking a mean-pool of the contextual token representations. The dialogue context is constructed by concatenating a maximum of three previous utterances and is then passed through SBERT encoder to obtain a fixed-size contextual dialogue representation.

\[
S = \text{MeanPooling}(\text{BERT}(S))
\]

In order to align the semantic vector space of the dialogue representations and the KG representations, we use SBERT to encode the KG elements. As KG entities and relations can be words or phrases, SBERT can effectively capture their semantic information. We use the publicly available SBERT-BERT-BASE-NLI\(^1\) model with mean-pooling as our SBERT encoder.

### 3.3 KG-CR\textsc{U}SE Architecture

KG-CR\textsc{U}SE learns to traverse a path on the KG by learning a function \(\pi_\theta\) that calculates the probability of an action \(a_t \in A_t\) given the state \(s_t\). The state \(s_t\) contains the dialogue history and entities already traversed by KG-CR\textsc{U}SE while decoding the paths, while \(a_t\) is the set of edges from the KG available to KG-CR\textsc{U}SE for extending its path.

The state \(s_t\) at step \(t\) is defined as a tuple \((D, (r_1, e_1, ..., r_{t-1}, e_{t-1}))\), where \(D\) is the dialogue context and \(r_i, e_i (i < t)\) are the relation and entity already decoded by KG-CR\textsc{U}SE at step \(i\). The initial state \(s_0\) is denoted as \((D, \emptyset)\), where \(\emptyset\) is the empty set.

At step \(t\), an action has the form \(a_t = (r_t, e_t) \in A_t\), where \(A_t\) is the set of all possible actions available to the model at step \(t\). \(A_t\) includes all outgoing edges of \(e_{t-1}\) in the KG \(\mathcal{G}\), i.e. \(A_t\) is the set of all the outgoing edges of the entity decoded by KG-CR\textsc{U}SE at timestep \(t - 1\). To let the agent terminate the search process, we add self-loop edges to every entity node in the graph denoting no operation (“self-loop”). The action \(a_t\) is represented as a concatenation of the relation and entity embedding \(a_t = [r_t; e_t]\), where \(r \in \mathbb{R}^{d_r}\), \(e \in \mathbb{R}^{d_e}\) and \(R_{d_r}, R_{d_e}\) are the size of the entity embedding and relation embedding respectively. At step 1, KG-CR\textsc{U}SE chooses between the entities mentioned in \(s_t\) for path traversal. The relation associated with action at step 1 is the zero vector. As mentioned, the state \(s_t\) contains the dialogue context and action history (path history). This sequential information in \(s_t\) is modelled using an LSTM:

\[
\begin{align*}
    d &= W_d D \\
    h_0 &= \text{LSTM}(0, d) \\
    h_t &= \text{LSTM}(h_{t-1}, a_{t-1}), t > 0
\end{align*}
\]

where \(D\) is the contextual dialogue embedding obtained using Equation 1 and \(W_d\) is a learnable matrix that maps the dialogue embedding to the LSTM input dimension. Given the hidden state representation \(h_t\) at time \(t\), KG-CR\textsc{U}SE assigns a probability to each action using Equation 6.

\[
x_t = W_{3, \theta}(\text{ReLU}(W_{2, \theta} h_t^1))
\]

\[
\pi_\theta(a_t | s_t, A_t) = \frac{\exp(a_t \cdot x_t)}{\sum_{a_t \in A_t} \exp(a_t \cdot x_t)}
\]

The hidden state representation \(h_t\) is passed through a two-layered dense network with ReLU activation (Nair and Hinton, 2010) in the first layer. The LSTM weights, \(W_{2, \theta} \in \mathbb{R}^{d_h \times d_h}\) and \(W_{3, \theta} \in \mathbb{R}^{(d_h + d_e) \times d_h}\) are the learnable parameters, and \(d_h\) is the LSTM hidden representation size.

### 3.3.1 Model Learning

We train KG-CR\textsc{U}SE by minimising the cross-entropy loss on the entities decoded at each timestep. Additionally, we train the model using teacher forcing (Sutskever et al., 2014), wherein the model makes each action conditioned on the gold history of the target path. To prevent overfitting, we add \(L_2\) regularisation to the parameters of the model. During training, we do not fine-tune the SBERT architectures, but back-propagate the gradients to the entity and relation embeddings.

### 3.3.2 KG-CR\textsc{U}SE Path Generation

Once the model is trained, KG-CR\textsc{U}SE takes the dialogue history and the entities mentioned in the current utterance as input, a horizon \(T\) and outputs a set of entity paths, relations paths of length \(T\) along with the probability score of each path. During inference, we remove self-loops from the KG except for the self-loop with label "self-loop". We do so to allow the agent traverse diverse paths rather than staying at entities mentioned in the dialogue history.
Table 1: Performance of KGDialAC in comparison with other baseline methods on different Recall@k metrics. The numbers reported are the mean values with the sample standard deviation (p=0.01). Results are statistically significant with p=0.01. Models with * denote our re-implementation.

<table>
<thead>
<tr>
<th>Model</th>
<th>path@1</th>
<th>path@5</th>
<th>path@10</th>
<th>path@25</th>
<th>tgt@1</th>
<th>tgt@5</th>
<th>tgt@10</th>
<th>tgt@25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tri-LSTM</td>
<td>3.2</td>
<td>22.6</td>
<td>36.3</td>
<td>56.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ext-ED</td>
<td>1.9</td>
<td>9.0</td>
<td>13.3</td>
<td>19.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Seq2Path</td>
<td>14.92</td>
<td>31.1</td>
<td>38.68</td>
<td>48.15</td>
<td>15.65</td>
<td>33.86</td>
<td>42.52</td>
<td>53.28</td>
</tr>
<tr>
<td>Seq2Seq*</td>
<td>6.53±0.78</td>
<td>26.21±1.21</td>
<td>35.02±1.27</td>
<td>45.78±1.18</td>
<td>7.13±0.85</td>
<td>30.64±1.62</td>
<td>41.01±1.43</td>
<td>52.97±1.55</td>
</tr>
<tr>
<td>DKGW*</td>
<td>14.16±1.16</td>
<td>37.26±1.91</td>
<td>47.85±2.60</td>
<td>59.20±2.33</td>
<td>14.96±1.04</td>
<td>39.53±1.81</td>
<td>51.06±2.15</td>
<td>63.85±1.58</td>
</tr>
<tr>
<td>AttnIO*</td>
<td>19.08±1.19</td>
<td>38.49±0.79</td>
<td>43.99±1.10</td>
<td>48.94±0.55</td>
<td>20.32±1.80</td>
<td>45.90±0.93</td>
<td>52.82±0.65</td>
<td>55.17±0.96</td>
</tr>
</tbody>
</table>

KG-CRUSE: 19.59±0.43 44.62±1.08 56.16±1.21 70.59±0.38 20.20±0.36 47.76±0.62 60.11±0.92 75.30±0.57

4 Experimental Setup

This section presents the dataset used, the baselines compared with and the description of the model settings of KG-CRUSE along with the metrics the models have been evaluated on.

4.1 Dataset

We evaluate our proposed framework on the OpenDialKG dataset (Moon et al., 2019). The dataset has 91,209 turns spread over 15,673 dialogues in the form of either task-oriented dialogues (recommendations) or chit-chat conversations on a given topic. Each turn is annotated with a KG path to represent fact transitions in the conversation. The KG is a subset of the Freebase KG (Bollacker et al., 2008) and has 1,190,658 fact triples, 100,813 entities and 1,358 relations. Following Moon et al. (2019), we split the dataset randomly into 70%, 15% and 15% for training, testing and validation.

4.2 Baselines and Evaluation Metrics

We compare KGDialAC against the following baseline models suggested by Moon et al. (2019) and Jung et al. (2020):

- Tri-LSTM (Young et al., 2018b): The model encodes all each utterance along with facts from the KG within 1-hop distance from the entities mentioned in the current utterance. This is used to retrieve facts from the KG for dialogue explanation.
- Ext-ED (Parthasarathi and Pineau, 2018b): Moon et al. (2019) conditioned the response generation with external knowledge vector input to generate response entity token at the final softmax layer, without using the structural information from the KG.
- Seq2Seq (Sutskever et al., 2014) model where the decoder is modified to generate entity paths by masking out unreachable nodes at each decoding step.
- Seq2Seq: A seq2seq (Sutskever et al., 2014) model where the decoder is modified to generate entity paths. Following Moon et al. (2019), we replace the softmax layer with a zero-shot learning layer in the KG embedding space.
- DKGW (Moon et al., 2019): A model to generate KG paths using domain-agnostic, attention-based recurrent graph decoder reinforced with a zero-shot learning layer over the KG embedding space.

Since the authors of OpenDialKG and AttnIO have not released their implementations, we report their performance on our re-implementations. We note that for most systems, our implementation is similar or better than the reported results. Regarding AttnIO, we were not able to reproduce the results although we note that errors in the implementation of the node sampler or leakage of the test dataset into the training dataset can easily lead to overestimation of the accuracy. We will release our re-implementations as well as our system as open-source code enabling future reproducibility.

We evaluate our models on different recall@k metrics for entity and path retrieval. Path@k measures if the ground-truth path is present in the top-k paths with the highest probability searched by the agent. Similarly, tgt@k measures if the response entity is present in the top-k entities retrieved by the agent. In situations where multiple paths point to the same response entity, we consider the path with the highest score for entity retrieval.
Table 2: Influence of sentence embeddings on KG-CRUSE performance. Comparison of different embedding methods.

<table>
<thead>
<tr>
<th>SBERT</th>
<th>Fine-tuned</th>
<th>P@1</th>
<th>P@25</th>
<th>E@1</th>
<th>E@25</th>
<th>Rel@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>No</td>
<td>12.74</td>
<td>66.72</td>
<td>12.98</td>
<td>72.14</td>
<td>39.37</td>
</tr>
<tr>
<td>ALBERT</td>
<td>Yes</td>
<td>13.42</td>
<td>65.67</td>
<td>13.96</td>
<td>72.23</td>
<td>40.93</td>
</tr>
<tr>
<td>S RoBERTa</td>
<td>No</td>
<td>17.17</td>
<td>68.04</td>
<td>17.65</td>
<td>73.34</td>
<td>40.71</td>
</tr>
<tr>
<td>SBERT</td>
<td>Yes</td>
<td>19.52</td>
<td>70.72</td>
<td>20.20</td>
<td>75.72</td>
<td>40.02</td>
</tr>
</tbody>
</table>

Table 3: Results on fine-tuning the SBERT architecture used for encoding the dialogue history. Additionally, the table reports the results of initialising the KG elements with random initialisation.

<table>
<thead>
<tr>
<th>SBERT</th>
<th>P@1</th>
<th>P@25</th>
<th>E@1</th>
<th>E@25</th>
<th>Rel@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>17.82</td>
<td>69.47</td>
<td>18.21</td>
<td>74.47</td>
<td>40.24</td>
</tr>
<tr>
<td>No</td>
<td>18.46</td>
<td>69.93</td>
<td>19.00</td>
<td>74.75</td>
<td>40.47</td>
</tr>
<tr>
<td>No</td>
<td>18.00</td>
<td>62.01</td>
<td>18.52</td>
<td>74.54</td>
<td>38.48</td>
</tr>
<tr>
<td>Yes</td>
<td>19.52</td>
<td>70.72</td>
<td>20.20</td>
<td>75.72</td>
<td>40.02</td>
</tr>
</tbody>
</table>

4.3 Implementation Details

For the task, we set horizon $T$ to 3. The dialogue, entity and relation embeddings are encoded using SBERT into a 768 dimensional vector. In KG-CRUSE, we consider 3 LSTM layers with $d_h = d_e + d_r = 1,536$. To prevent the agent from overfitting on the dataset, we add $L_2$ regularisation with a weight decay parameter of 1e-3.

Similar to Jung et al. (2020), we set the batch size to 8 and train the model with Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1e-4 for 20 epochs. We report the results on five different splits of the data. As entity occurrences in a dialogue dataset are sparse, it is desirable to report the performance on five different splits of the data rather than an assessment of five models on one split.

5 Results and Discussion

We begin with performing a quantitative evaluation of the models. Following this, we study the impact of our choice of sentence embeddings on the model performance. Then we analyse the impact of beam-width at each decoding step during inference. Finally, we provide insights of examples where the results of KG-CRUSE are not consistent with the ground truth paths.

5.1 Quantitative Analysis

In this section, we compare the performance of our proposed approach against the different baselines. From Table 1, it can be observed that KG-CRUSE performs better than the different baseline models on the OpenDialKG dataset. For entity and path accuracy, AttnIO has the closest performance compared to our model, with the latter being 2.7% relatively better on both path@1 and entity@1 metrics. On increasing $k$ of recall@$k$, we find KG-CRUSE has at least 10% relative improvement over the baseline models. It is interesting to notice that on increasing the value “k”, KG-CRUSE performs relative better than other models. KG-CRUSE identifies paths semantically relevant to the dialogue context although different from the gold-label paths. The huge gain on the path@25 metric advocates for this hypothesis. It is worthwhile to notice that although AttnIO has the closest performance for path@1 and entity@1 to KG-CRUSE, the performance degrades when “k” increases in path@k and entity@k. This might be due to the fact that the beam-width reported by the authors is not expressive enough to capture semantically relevant paths or entities.

5.2 Effectiveness of Sentence Embeddings

In our framework, we utilise sentence SBERT embeddings to encode dialogue context and KG elements. In this section, we conduct an ablation study on the efficacy of such embeddings. We replace the SBERT embeddings with the [CLS] token representation of BERT$^2$ (Devlin et al., 2019) and ALBERT model$^3$ (Lan et al., 2020) in KG-CRUSE. Additionally, to show the effectiveness of our choice of embeddings, we consider an instance wherein the elements are encoded using Sentence-RoBERTa (S RoBERTa)$^4$ (Reimers and Gurevych, 2019). The results in Table 2 demonstrate the strength of our embedding choices wherein SBERT/S RoBERTa outperforms the BERT and ALBERT embeddings. Both the sentence embedding models models are pre-trained on NLI datasets, which allows them to capture rich semantic information for textual similarity. These embeddings have demonstrated strong performances in the task of semantic search using cosine-similarity (Reimers and Gurevych, 2019). It should be noted that before the softmax layer in KG-CRUSE, we compute the dot product of the LSTM layer hidden representation with that of the relation-entity embeddings available at the given timestep. As a result of this step, we expect the

---

1. https://huggingface.co/bert-base-uncased
3. https://huggingface.co/albert-base-v2
performance of SBERT/SRoBERTa embeddings to be better that BERT/ALBERT embeddings.

Additionally, we see from the Table 2 that the relation accuracy of different models is higher than path accuracy. This is due to the outgoing edges of a node (from the dialogue history) sharing similar features if they are connected using the same relation. Thus multiple entities can fit our choice of the response entity given the dialogue context.

5.3 Impact of KG Embedding Alignment and SBERT Fine-tuning

In this section, we study the impact of encoding KG elements with SBERT embeddings. Additionally, we analyse if fine-tuning the SBERT architecture used for encoding the dialogue history is beneficial for explainability.

Table 3 outlines four situations, where in two cases we fine-tune the SBERT architecture used for encoding the dialogue history. We also consider two cases where the embeddings of the KG elements are initialised with values drawn from a normal distribution with mean 0 and standard deviation 1, corresponding to the value “No” in the second column. It should be noted that we never consider fine-tuning the SBERT architecture used for encoding the KG elements.

We see from the Table 3 that in cases when the KG elements are not encoded with SBERT embeddings, their performance drops as compared to cases when we use SBERT embeddings. Additionally, we find that fine-tuning SBERT leads to a decrease in the performance of KG-CRUSE. This can be attributed to the change in semantic space of the dialogue embeddings and the KG embeddings. Hence, we do not finetune the SBERT architecture in the default setting of KG-CRUSE.

5.4 Impact of Beam-Width on Path Reasoning

In this experiment, we study the influence of beam-width at different timesteps on the model performance. The first column of Table 4 lists the tuples $K_1$, $K_2$, $K_3$ where each $K_i$ denotes the top-$K_i$ edges sampled at timestep $i$.

We conduct this analysis on a single split of the dataset keeping all other parameters of the model constant. We consider a diverse set of values for each $K_i$. From Table 4, we find that although the tuples (2, 5, 50), (2, 10, 25), (2, 25, 10) and (2, 50, 5) have an equal number of sampled paths, tuple (2, 25, 10) performs better than others. Interestingly, we observe that the sampling sizes at the second timestep play a significant role in finding optimal paths for KG-CRUSE. The first two sets of selection of facts (i.e. during timesteps 1 and 2) largely determine the facts reachable by KG-CRUSE. Sampling more samples during the initial timesteps enables the agent to explore diverse paths initially and KG-CRUSE then makes an optimal selection of facts dependent on the dialogue information.

5.5 Analysis of Computational Requirements

In this study, we conduct an analysis of the training time required for training the model. We also compare the performance of different architectures with regards to the inference speed.

Table 5 shows that while DKGW has a better train time per epoch than KG-CRUSE and Seq2Seq has a better inference speed than KG-CRUSE, we can observe from Table 1 that our model achieves better performance compared to these models. It is worthwhile to mention that while AttnIO achieves the closest performance to KG-CRUSE as shown in Table 1, it requires roughly six times more training time and is ten times slower during inference. This clearly indicates the benefits of using KG-CRUSE for explain-
This section highlights three scenarios showcasing conversational reasoning. We utilise SBERT embeddings to capture the rich semantic information in the dialogue history and the KG elements. We conduct an extensive evaluation to demonstrate that our framework outperforms several baseline models on both explainability and response entity retrieval. As annotating ground truth demonstrations is expensive, we plan on extending this model to scenarios involving the generation of dialogue-conditioned expert demonstrations.

### 6 Conclusion

In this work, we propose KG-CRUSE, an LSTM based lightweight framework for explainable conversational reasoning. We utilise SBERT embeddings to capture the rich semantic information in the dialogue history and the KG elements. We conduct an extensive evaluation to demonstrate that our framework outperforms several baseline models on both explainability and response entity retrieval. As annotating ground truth demonstrations is expensive, we plan on extending this model to scenarios involving the generation of dialogue-conditioned expert demonstrations.

### References


Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and...


