FALCON: A Faithful Contrastive Framework for Response Generation in TableQA Systems

Shineng Fang[1], Jiangjie Chen[1], Xinyao Shen[1], Yunwen Chen[3], and Yanghua Xiao[1,2(✉)]

[1] Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, Shanghai, China
{snfang19,jjchen19,xinyaoshen19,shawyh}@fudan.edu.cn
[2] Fudan-Aishu Cognitive Intelligence Joint Research Center, Shanghai, China
[3] DataGrand Inc., Shanghai, China
chenyunwen@datagrand.com

Abstract. In a practical TableQA system, response generation is a critical module to generate a natural language description of the SQL and the execution result. Due to the complex syntax of SQL and matching issues with table content, this task is prone to produce factual errors. In this paper, we propose FALCON, a FAithfuL CONtrastive generation framework to improve the factual correctness of generated responses. FALCON forces the generation model to identify examples with factual errors in the latent space during training and takes contrastive examples into consideration during inference. We also propose two new automatic metrics to further evaluate faithfulness specialized to this task. Experimental results show FALCON brings a favorable performance improvement on both automatic and human evaluation amongst various baseline methods (The code of FALCON is released at https://github.com/whuFSN/FalCon).

Keywords: Response generation · Factual correctness · Contrastive learning

1 Introduction

With extensive research on the Natural Language Interface of Databases (NLIDB), semantic parsing task has made significant progress in recent years [4,12,27,31]. However, in a practical TableQA system [37], *response generation* (RG) is also a critical module to interact with users' natural language questions. As shown in Fig. 1, a practical TableQA system consists of semantic parsing and response generation module. RG requires generating a natural language description of the SQL and the execution result. In general, we argue that such a response generation module is necessary due to the following two reasons: 1) The generated response could help users verify whether the query result is consistent with the original question; 2) The generated response provides a concise and easy-to-understand summary about the result table.

However, previous SQL-Interface-related studies rarely investigate RG in any systematic way. Previous work [3,13,33] mainly focus on the SQL-to-Question task, which generates natural language descriptions interpreting the meaning of a given SQL. Different from SQL-to-Question, the unique challenges of RG lie in: 1) In addition to the

A. Bhattacharya et al. (Eds.): DASFAA 2022, LNCS 13247, pp. 197–212, 2022.
https://doi.org/10.1007/978-3-031-00129-1_13

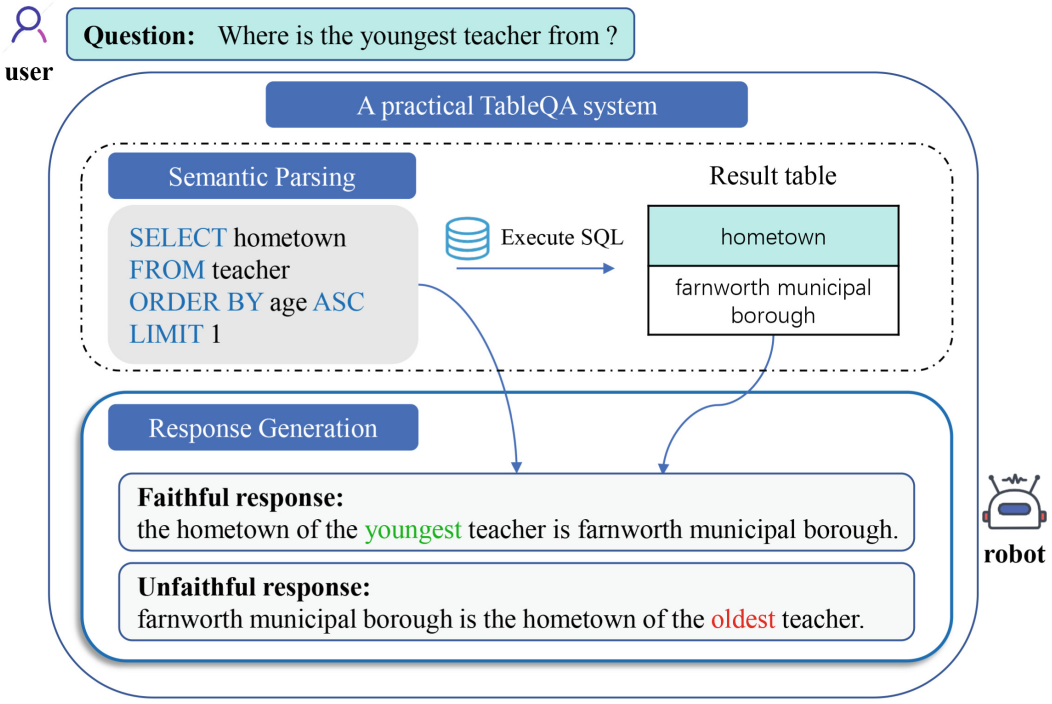

Fig. 1. An illustration of a TableQA system, which consists of semantic parsing and response generation module. RG takes the SQL and result table as input to generate the response. The unfaithful response is highlighted in red, which is not factually consistent with the input. (Color figure online)

SQL, RG also takes the execution result as input, which needs an explanation of table content; 2) The benchmark of RG is CoSQL [35], which includes more complex SQL grammar (e.g., nested queries, multi-table queries).

Besides fluency and grammaticality, factual correctness is also an important factor when evaluating the quality of generated responses. While advanced Natural Language Generation (NLG) techniques have been successful in producing fluent text, these methods still face great challenge caused by factual incorrectness [11,17]. As shown in Fig. 1, an unfaithful response is easily generated by T5 [26], a popular pre-trained generation model, which misunderstands the "*youngest teacher*" as the "*oldest teacher*". This error implies that the generic T5 is unaware of sorting by age (in ascending order) specified in the input SQL. Unfortunately, such an error is not accidentally generated. Our new task has several inherent properties that make models prone to produce factual errors. First, responses contain rich logical information, which is mainly specified in aggregate functions (e.g., `MAX`), logical operators (e.g., $<$), and abstract fields (e.g., `dept_name`) in SQL. Second, content matching between SQL and the table is also a challenge.

In this paper, we propose a FAithfuL CONtrastive generation framework (FALCON) to improve factual correctness of generated responses. Contrastive learning [6,24,30] learns representations by contrasting positive pairs and negative pairs. We intentionally introduce contrastive learning into RG to handle factual errors. Specifically, we utilize a series of heuristic rules to construct samples with factual errors which we

call *imposters*. During training, in addition to optimizing naive training objective, our models attempt to distinguish between imposters and ground truths in the latent space. During inference, an imposters-contrastive decoding method is proposed to avoid generating unfaithful responses.

We also proposed two metrics to evaluate the faithfulness of generated responses. Previous research [10] has shown that ROUGE [20], a widely used metric based on n-gram overlap, is not always a valid factuality metric. This motivates us to propose two new faithfulness-aware metrics. The first metric is *Logical Score*, which mainly reflects the logical matching degree between the generated response and input SQL. Specifically, we parse the generated response into SQL components by a trained Response-to-SQL model. Then Logical Score is obtained by calculating the matching degree between parsed SQL and input SQL. The second metric is *Consistency Score*, which is estimated by a consistency classifier trained on positive and negative samples. This metric represents the consistency between the generated response and the input.

In brief, our contributions can be summarized as follows:

- We are the first work to formulate the response generation task in a TableQA system. Focusing on factual correctness, we propose a faithful contrastive generation framework (FALCON) to boost the faithfulness of responses.
- Two metrics, *Logical Score* and *Consistency Score*, are proposed to evaluate the factual correctness of generated responses. Experiments show they have a high correlation with human judgment.
- Extensive evaluations demonstrate the state-of-the-art (SoTA) performance of FALCON on both automatic and human evaluation.

2 Related Work

SQL-to-Question. A natural language generation task that generates a natural language description from the SQL. Iyer *et al.* [14] employ LSTM networks with attention to generate descriptions of the SQL which regards the SQL as a sequential text. Xu *et al.* [33] represent the SQL as a directed graph and use a Graph2seq model to encode this graph-structured SQL. Cai *et al.* [3] leverage the syntactic structure of abstract syntax trees (AST) to encode the SQL with a novel type-associated encoder.

Factual Correctness. Previous work on factual correctness focus on the summarization task. Cao *et al.* [5] propose to improve summarization models by leveraging open information extraction. Falke *et al.* [9] apply natural language inference to evaluate the factual correctness of the generated summaries. Zhang *et al.* [39] utilize reinforcement learning to optimize the faithfulness of radiology summaries, whose reward is obtained from a trained fact extractor. In the Data-to-Text task, Wang *et al.* [32] generate a faithful table description by introducing a new table-text optimal-transport matching loss and a table text embedding similarity loss.

Contrastive Learning. Contrastive learning is to learn representations by contrasting positive pairs and negative pairs. Contrastive representation learning builds a generative model to score positive samples higher than negative samples in computer vision

[6,24]. Contrastive learning also has many applications in natural language processing. To learn a sentence's representation, Lajanugen *et al.* [23] take its consecutive sentences as positive samples and sentences from another document as negative samples. By assigning a higher probability to a ground-truth translation than erroneous translation, Yang *et al.* [34] apply contrastive learning to reducing word omission errors in machine translation. The gradient-based method [18], which constructs positive and negative pairs automatically, is proposed to tackle the exposure bias problem.

3 Task Definition and Preliminary

In this section, we formally define the response generation task, and then we describe the backbone model under Sequence-to-Sequence (Seq2Seq) architecture to tackle the task.

Given the input X including a SQL S and a result table T, our goal is to generate the corresponding response Y. The SQL S is represented as a token sequence $S = s_1, s_2, ..., s_{|S|}$. The table $T = \{T_{i,j} | 1 \leq i \leq R_T, 1 \leq j \leq C_T\}$ has R_T rows and C_T columns with each $T_{i,j}$ being the content in the (i, j) cell. $T_{i,j}$ could be a word, a phrase or a number. The annotated response is $Y = y_1, y_2, ...y_{|Y|}$. We aim to train a response generator $p(Y|X)$ to generate response from X. The generated response is expected to be both *fluent* and *faithful*.

Modeling the response generation task under Seq2Seq paradigm, we choose T5 [26], a strong pre-trained generation model as our backbone model. In fact, FALCON can also be applied to other Data-to-Text methods, as shown in our experiments. Following previous work on linearizing table as natural language [7], we use template to flatten the table T as a paragraph $T_L = t_1, ..., t_{|T_L|}$. If the result table is empty, we use "No, there is no result" to represent the result. Then we concatenate SQL S and linearized table T_L as a complete sequence input $X = x_1, ..., x_{|X|}$.

A typical approach to train T5 is to minimize the negative log-likelihood (NLL) of the target sequence Y, which we refer to as $\mathcal{L}_{nll}$.

$$\begin{aligned} \mathcal{L}_{nll} &= -\sum_{t=1}^{|Y|} \log p(y_t | X, y_{<t}) \\ p(y_t | X, y_{<t}) &= \texttt{softmax}(\mathbf{W}\mathbf{h}_t^D + \mathbf{b}) \\ \mathbf{h}_t^D &= \text{Decoder}(y_{<t}, \mathbf{H}^E) \\ \mathbf{H}^E &= \text{Encoder}(X) \end{aligned} \tag{1}$$

where $\mathbf{H}^E = [\mathbf{h}_1^E, ..., \mathbf{h}_{|X|}^E] \in \mathbb{R}^{d \times |X|}$ is the concatenation of hidden state of input tokens and $\mathbf{H}^D = [\mathbf{h}_1^D, ..., \mathbf{h}_{|Y|}^D] \in \mathbb{R}^{d \times |Y|}$ is the concatenation of hidden state of output tokens. d is the hidden size of T5.

4 Proposed Approach

First, we describe the method of constructing imposters, which are crucial for the performance of FALCON. Then we introduce our faithful contrastive generation framework

Table 1. Examples of constructing imposters through text transformations. Blue and red text highlight the changes made by the transformation.

Transformation	Original sentence	Transformered sentence
Construct $\widetilde{X}$	`SELECT f_id FROM files WHERE formats = "mp4" UNION SELECT f_id FROM song WHERE resolution > 720`	`SELECT f_id FROM files WHERE formats = "mp4" EXCEPT SELECT f_id FROM song WHERE resolution > 720`
	`SELECT address FROM member WHERE age < 30`	`SELECT address FROM member WHERE age > 30`
	`SELECT MAX ( Length ) FROM roller_coaster`	`SELECT MIN ( Length ) FROM roller_coaster`
	`SELECT name FROM country ORDER BY surfacearea DESC LIMIT 1`	`SELECT name FROM country ORDER BY surfacearea ASC LIMIT 1`
	`SELECT AVG ( salary ) FROM instructor`	`SELECT AVG ( salary ) FROM dept_name`
Construct $\widetilde{Y}$	The number of drivers who are from Hartford City or younger than 40 is 11	The number of drivers who are from Hartford City or older than 40 is 11
	The average damage for all storms is 11.0629 million USD	The maximum damage for all storms is 11.0629 million USD
	The average damage for all storms is 11.0629 million USD	The average storm for all storms is 11.0629 million USD

(FALCON) as shown in Fig. 2, which consists of two major components: contrastive training and contrastive inference. Finally, we elaborate on two metrics to evaluate faithfulness.

4.1 Imposters Construction

To make the model subject to factual constraints when learning to generate target sequence, we construct negative samples that are syntactically identical but have significant differences in factual correctness, which we call imposters.

Given a pair (X, Y), we construct imposters through a series of predefined transformations. To be specific, source imposter $\widetilde{X}$ is obtained by modifying original input X, and original output Y is modified to yield target imposter $\widetilde{Y}$. Consequently, a tuple $(X, Y, \widetilde{X}, \widetilde{Y})$ is constructed to be applied to FALCON. As shown in Table 1, we utilize following heuristic rules to construct imposters.

Construct Source Imposter $\widetilde{X}$: Some keywords play the same role in SQL syntax. When the original keyword is replaced by other keywords, the query intention or condition of SQL will change. Apparently, the modified SQL is factually inconsistent with the original output. Specifically, 1) replace the `UNION`, `INTERSECT`, and `EXCEPT` with another one of the above keywords. e.g., `UNION` to `EXCEPT`; 2) replace comparison

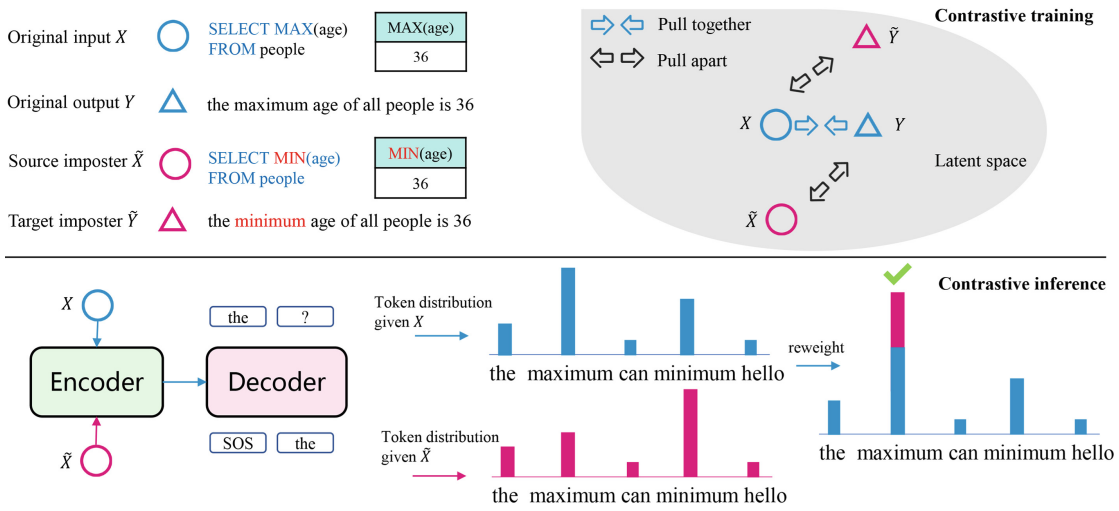

Fig. 2. An illustration of FALCON, which consists of contrastive training and contrastive inference. In contrastive training, the backbone model is enforced to pull factually consistent pairs (X, Y) together and push the factually inconsistent pairs $(\widetilde{X}, Y), (\widetilde{X}, Y)$ apart in the latent space. In contrastive inference, FALCON generates a more faithful response by referring to the token probability distribution of source imposter $\widetilde{X}$.

operators ($>, <, \geq, \leq, !=$) with another comparison operators. e.g., $>$ to $<$; 3) replace aggregation keywords (`MAX`, `AVG`, `MIN`, `COUNT`, `SUM`) with another aggregation keywords. e.g., `MAX` to `MIN`; 4) replace order keywords (`ASC`, `DESC`) with another order keywords. e.g., `DESC` to `ASC`; 5) use string-match based method to recognize columns and tables mentioned in the response, then replace it with another column or table[1].

Construct Target Imposter $\widetilde{Y}$: We modify the logical expression involved in the response so that the modified response is inconsistent with the original input. Specifically, 1) use string-based methods to locate comparative and superlative and replace them with antonyms. e.g., younger to older; 2) replace the tokens that represent aggregate keywords. e.g., average to maximum; 3) replace the span mentioned in SQL or table with another randomly sampled span.

4.2 Contrastive Training

We argue that the generation model should be restricted by factual consistency in the latent space. To achieve this, FALCON encourages the latent semantics of the ground truth response to be consistent with the SQL and table. Specifically, given a pair of input and output (X, Y), we first project the original input and output to the latent space:

$$\begin{aligned} \mathbf{h}^E &= \texttt{MeanPool}(\texttt{Relu}(\mathbf{W}^E\mathbf{H}^E + \mathbf{b}^E)) \\ \mathbf{h}^D &= \texttt{MeanPool}(\texttt{Relu}(\mathbf{W}^D\mathbf{H}^D + \mathbf{b}^D)) \end{aligned} \tag{2}$$

[1] Synchronously modify headers and values in the result table.

where $\mathbf{h}^E, \mathbf{h}^D \in \mathbb{R}^d$ represent the latent representation of input and output. $\mathbf{W}^E$ and $\mathbf{W}^D$ are the source and target projection matrix, respectively. `MeanPool` denotes average pooling operation. Then we calculate their consistency $\mathcal{C}(X, Y)$ in the latent space:

$$\mathcal{C}(X, Y) = \sigma(\mathbf{W}^C[\mathbf{h}^E; \mathbf{h}^D] + b^C)) \tag{3}$$

where $\mathbf{W}^C \in \mathbb{R}^{2*d}$ and $b^C \in \mathbb{R}$ are learnable parameters. σ is the sigmoid function and $[\cdot;\cdot]$ denotes the operator of vector concatenation.

Models not only learn from faithful examples but also benefit from identifying imposters. Hence, given a tuple $(X, Y, \widetilde{X}, \widetilde{Y})$ where $\widetilde{X}$ and $\widetilde{Y}$ are the source and target imposter, FALCON enforces the model to provide a lower consistency score for imposters than ground truth.

For the target imposter $\widetilde{Y}$, we first compute its representation $\widetilde{\mathbf{h}}^D$ in the latent space given original input X:

$$\begin{aligned} \widetilde{\mathbf{h}}^D &= \texttt{MeanPool}(\texttt{Relu}(\mathbf{W}^D\widetilde{\mathbf{H}}^D + \mathbf{b}^D)) \\ \widetilde{\mathbf{h}}^D_t &= \text{Decoder}(\widetilde{y}_{<t}, \mathbf{H}^E). \end{aligned} \tag{4}$$

Then the consistency $\mathcal{C}(X, \widehat{Y})$ between original input X and target imposter $\widetilde{Y}$ is obtained. According to the principle that a consistent pair should be assigned a higher score, we adopt margin ranking loss [1] to separate original output and target imposter.

$$\begin{aligned} \mathcal{L}_t &= \max(0, \delta - \mathcal{C}(X, Y) + \mathcal{C}(X, \widetilde{Y})) \\ \mathcal{C}(X, \widetilde{Y}) &= \sigma(\mathbf{W}^C[\mathbf{h}^E; \widetilde{\mathbf{h}}^D] + b^C)) \end{aligned} \tag{5}$$

where margin δ is the hyper-parameter.

For the source imposter $\widetilde{X}$, we obtain its latent representation $\widetilde{\mathbf{h}}^E$ through the encoder. Then FALCON requires the model to distinguish between original input and source imposter in the latent space.

$$\begin{aligned} \mathcal{L}_s &= \max(0, \delta - \mathcal{C}(X, Y) + \mathcal{C}(\widetilde{X}, Y)) \\ \mathcal{C}(\widetilde{X}, Y) &= \sigma(\mathbf{W}^C[\widetilde{\mathbf{h}}^E; \mathbf{h}^D] + b^C)) \\ \widetilde{\mathbf{h}}^E &= \texttt{MeanPool}(\texttt{Relu}(\mathbf{W}^E\widetilde{\mathbf{H}}^E + \mathbf{b}^E)) \\ \widetilde{\mathbf{H}}^E &= \text{Encoder}(\widetilde{X}). \end{aligned} \tag{6}$$

The overall optimization object is as follows:

$$\mathcal{L} = \mathcal{L}_{nll} + \alpha(\mathcal{L}_s + \mathcal{L}_t) \tag{7}$$

where the contrastive weight α controls the relative importance of contrastive learning.

4.3 Contrastive Inference

A typical decoding approach is that when decoding the t-th step, we compute the probability distribution of each token on target vocabulary $\mathcal{V}$. $p(y_t^k|X, y_{<t})$ represents the probability of the k-th token $\mathcal{V}_k$ on vocabulary at step t:

$$p(y_t^k|X, y_{<t}) = p(y_t = \mathcal{V}_k|X, y_{<t}). \tag{8}$$

We argue that referring to the token probability distribution of source imposter $\widetilde{X}$ can help improve faithfulness. Therefore, we propose an imposters-contrastive decoding method to generate more faithful responses. We first obtain the imposter $\widetilde{X}$ of the original input X through the construction method[2] mentioned in Subsect. 4.1. Intuitively, we should encourage tokens to be generated based on the original input, while suppressing the token probability generated by the source imposter.

Then we compute the proportion of the token $\mathcal{V}_k$ probability generated by the original input X compared to the source imposter $\widetilde{X}$:

$$\alpha_t^k = \frac{\exp p(y_t^k|X, y_{<t})}{\exp p(y_t^k|X, y_{<t}) + \exp p(y_t^k|\widetilde{X}, y_{<t})}. \tag{9}$$

By setting a threshold of $\rho > 0.5$, for token $\mathcal{V}_k$ with α_t^k greater than ρ, it means that it's generated specifically based on the original input compared to imposter, which should be encouraged. Therefore, we increase the probability of these tokens to generate words that are more consistent with the original input:

$$\delta_t^k = \begin{cases} \lambda \alpha_t^k p(y_t|X, y_{<t}), & \alpha_t^k \geq \rho \\ 0, & \alpha_t^k < \rho \end{cases} \tag{10}$$

where λ is the hyper-parameter to control the impact of imposter on responses generation. δ_t^k is the probability increment.

The final probability of the k-th token is computed as:

$$p'(y_t^k|X, \widetilde{X}, y_{<t}) = p(y_t^k|X, y_{<t}) + \delta_t^k. \tag{11}$$

4.4 Metrics

In this subsection, we introduce two metrics inspired by two principles to evaluate faithfulness of generated responses.

Principle 1. *The generated response should contain the logical information of the input SQL, including the query intent and condition.*

Logical Score. According to *principle 1*, we train a model to parse the response into SQL. Through this model, the generated response is translated into various components of SQL. Then we conduct a fine-grained matching between the parsed SQL and the input SQL to compute its logical accuracy. Specifically, we adopt BRIDGE [21], the SoTA Text-to-SQL semantic parser on Spider [36], as our Response-to-SQL model. Following previous Text-to-SQL evaluation methods [40], we employ Exact Match as the logical score to measure the logical accuracy between the generated response and the original SQL.

Principle 2. *A model that can distinguish between ground truth and imposter should be able to evaluate the correctness of the generated response.*

[2] We only use the first three rules for constructing source imposters when inference.

Consistency Score. According to *principle 2*, we build a consistency classifier to score the generated response. Given a tuple $(X, Y, \widetilde{X}, \widetilde{Y})$ constructed from Sect. 4.1, (X, Y) is a consistent pair as our positive example. $(\widetilde{X}, Y)$ and $(X, \widetilde{Y})$ are inconsistent pairs as negative examples. A RoBERTa-based [22] consistency classifier is trained to distinguish whether the response is factually consistent with the given SQL and result. Specifically, taking the concatenation of SQL, linearized table, and response as input, this classifier predicts whether the generated output is a faithful statement about the input. We employ the average predicted score as the consistency score.

5 Experiments

In this section, we conduct several experiments to demonstrate the effectiveness of our method.

5.1 Dataset

CoSQL. We evaluate FALCON on CoSQL [35], consisting of 7845/1074 examples for train/development. Each example includes the SQL, the result table, and the corresponding response. Different from WikiSQL [40], the SQL in CoSQL have following complex syntax: 1) multi-table queries; 2) nested queries; 3) advanced keywords (`HAVING`, `ORDER BY`, `UNION`, `DISTINCT`, `LIMIT`, etc.)

5.2 Baselines

We apply FALCON to several competitive generation models. Due to the lack of relevant baselines, we extend some Data-to-Text methods to adapt to this task. These methods can be divided into two categories, one is structure-based methods, and the other is textual-based methods. Structure-based methods can explicitly encode the internal structure of SQL and table to generate high-quality text.

- **GraphWriter** [16]: A Graph-to-Text generation model that utilizes a graph network to encode the knowledge graph (KG) and RNN to encode its title. We parse the SQL into an Abstract Syntax Tree (AST) as the input KG and take the linearized table as its input title.
- **GTN** [2]: A Graph Transformer that uses explicit relation encoding and allows direct communication between two distant nodes. Following [41], we convert the result table into a graph. Then we use a global node to connect it to the AST as the entire input graph.

Based on the Seq2Seq architecture, textual-based methods take the sequential SQL and the linearized table as input to generate the response.

- **Transformer** [29]. A representative model of neural machine translation. We add the copy mechanism to enable the model to copy words from the input.
- **BART** [19]: A pre-trained language model (PLM) which is pre-trained as a text-to-text denoising autoencoder. We use BART-base in this paper.
- **T5** [26]: A PLM which converts every language problem into a Text-to-Text format. We use T5-small in this paper.

5.3 Implementation Details

First, we try our best to adjust the hyper-parameters so that the backbone model has a relatively optimal performance. Then according to the same hyper-parameter settings, we apply FALCON to improve the factual correctness. Taking backbone model T5 as an example, we optimize models by Adam [15] with a learning rate of 5e−5 and the batch size of 32. The margin δ is set to 0.6 and α is set to 5.0. During inference, the greedy search is adopted to decode the response and the maximum decoding length is 25. λ and ρ are set to 0.2 and 0.8, respectively.

To train the Response-to-SQL model, we follow the same hyper-parameter settings as [21] except that the batch size is changed to 12. In order to filter out trivial responses like "yes", we discard the top 20% of the samples with the lowest edit distance between its SQL and response in the dataset. We also add Question-to-SQL examples from spider [36] to train set because we found it can significantly improve its performance. Finally, its exact match accuracy on the development set is 61.7%.

We train the consistency classifier with RoBERTa-base [22]. The synthetic train set includes 7,845 positive samples and 13,250 negative samples. 1,074 positive samples and 1,778 negative samples are included in the synthetic development set. On the synthetic development set, the classifier has an 86.5% F1-score.

5.4 Evaluation Metrics

Following the common practice, we illustrate the n-gram based BLEU [25] and ROUGE-L [20] evaluations to evaluate the quality of our generated response. We also evaluate the results using BERTScore [38] and BLEURT metric [28], which employ contextual embeddings to incorporate semantic information. Logical Score and Consistency Score are also used to evaluate the faithfulness of generated responses.

We invited three graduate students majoring in CS as experiment participants. Each of them is familiar with SQL. To ensure accurate human evaluation, the raters are trained with word instructions and text examples of the grading standard beforehand. Specifically, Two annotators are first asked to evaluate fluency and faithfulness for 600 samples separately. Then they re-check each other's results. If two annotators have conflicts, we will send these samples to the third annotator for final judgment. The following are detailed criteria for evaluating fluency and faithfulness.

Fluency

- **0**: The response is smooth and natural.
- **1**: The response does not flow smoothly but people can understand its meaning.
- **2**: The response is not fluent at all.

Faithfulness

- **0**: The response has no factual errors.
- **1**: The response has a factual error.
- **2**: The response has more than one factual error.

Table 2. Evaluation results of the different models on automatic metrics. "+FALCON" means the model is trained using FALCON. L-Score and C-Score denote Logical Score and Consistency Score, respectively. BLEURT-T and BLEURT-B denote BLEURT-Tiny and BLEURT-Base. The best results in each group are highlighted in **bold**.

	BLEU	ROUGE	BERTScore	BLEURT-T	BLEURT-B	L-Score	C-Score
GraphWriter	16.86	47.44	37.42	−27.68	−58.43	28.70	55.04
Graphwriter+FALCON	**17.06**	**48.39**	**38.30**	**−24.28**	**−57.30**	**30.48**	**55.26**
GTN	18.31	**51.55**	46.68	0.20	−27.95	35.46	66.49
GTN+FALCON	**18.70**	51.48	**47.34**	**1.53**	**−25.84**	**37.19**	**67.81**
Transformer	12.88	43.42	32.05	−23.52	−66.50	17.61	43.52
Transformer+FALCON	**13.13**	**43.62**	**32.60**	**−21.98**	**−63.96**	**17.89**	**43.76**
BART	24.60	57.39	58.72	22.38	9.28	54.10	83.96
BART+FALCON	**24.73**	**57.51**	**59.12**	**23.36**	**9.60**	**54.79**	**84.52**
T5	25.25	57.54	57.89	22.40	6.74	53.79	85.46
T5+FALCON	**25.65**	**57.89**	**58.41**	**23.92**	**7.76**	**54.32**	**85.58**

6 Result and Analysis

6.1 Performance on Automatic Metrics

Table 2 presents the automatic evaluation results of different backbone models with conventional NLL training and FALCON[3]. On the whole, applying FALCON to different models consistently improves the factual correctness of generated responses.

Specifically, FALCON slightly improves the performance of models on BLEU and ROUGE. This is reasonable because these metrics measure the n-gram overlap between a reference response and candidate response, which are not always valid factuality metrics [10]. Even so, applying FALCON to GraphWriter improve 0.20 on BLEU and 0.95 on ROUGE.

FALCON has achieved noticeable improvements on BERTScore and BLEURT. Such PLM-based metrics reflect the factual correctness by matching semantic information in the embedding space [10]. FALCON has achieved an average improvement of 1.42 points on BLEURT-Base, which is a favorable performance boost. These improvements demonstrate that by encouraging the model to distinguish well-designed contrastive pairs, the model learns a more robust representation and produces responses that are more faithful to the input.

Applying FALCON has improved each model on Logical Score and Consistency Score. FALCON has achieved an average improvement of 1.00 points on Logical Score and 0.49 points on Consistency Score. The improvement of Logical Score shows that FALCON has produced a response that is more logically consistent with the input SQL. The enhancement of Consistency Score indicates that the response generated by FALCON is more consistent with the input.

[3] We report the average best performance observed in 3 runs on the development set of CoSQL since its test set are not public. All improvements of FALCON are significant with $p < 0.01$ compared to backbone models.

Table 3. Human evaluation on fluency and faithfulness. Both metrics are the smaller the better.

	Fluency	Faithfulness	Kappa
GraphWriter	1.23	1.32	0.69
Graphwriter+FALCON	**1.02**	**0.98**	0.72
GTN	0.86	0.84	0.64
GTN+FALCON	**0.71**	**0.65**	0.53
T5	0.45	0.52	0.61
T5+FALCON	**0.35**	**0.27**	0.71

Table 4. Ablation study of our framework components using T5. "C-inference" means using contrastive inference, "S-imposter" means using source imposters, "T-imposter" means using target imposters, and "C-training" means training T5 with both source and target imposters.

	BLUERT-T	BLEURT-B	L-Score	C-Score
(a) T5	22.40	6.74	53.79	85.46
(b) *w/ C-inference*	22.78	6.90	54.23	85.55
(c) *w/ S-imposter*	22.97	**7.78**	54.03	**85.61**
(d) *w/ T-imposter*	22.59	7.48	53.91	85.27
(e) *w/ C-training*	23.41	7.66	54.12	85.46
(f) Full version	**23.92**	7.76	**54.32**	85.58

6.2 Performance on Human Evaluation

FALCON *Performance* We further conduct extensive human evaluations of generated responses. Generation results of six models on 100 samples are provided to three students for blind testing to evaluate the fluency and faithfulness.

Table 3 presents evaluation results and Cohen's kappa scores [8] to measure the intra-rater reliability. With FALCON, T5 has reduced the average number of factual errors per response from 0.52 to 0.27. In general, applying FALCON to GTN and T5 has achieved significant improvement in the faithful aspect, which demonstrates our framework's effectiveness to boost factual correctness.

Metrics Performance. We also measure the correlation (Pearson's r) between the faithfulness and two new metrics. The correlation coefficient between Logical Score and faithfulness is -0.68 (with $p < 0.05$), and -0.71 (with $p < 0.05$) is the correlation coefficient of Consistency Score. This result reveals they have a relatively high correlation with human judgment on faithfulness, which illustrates their effectiveness to evaluate the faithfulness of responses.

6.3 Model Analysis

Effect of Contrastive Training. To constrain the model with factual correctness, we require the model to distinguish ground truths from imposters in the latent space when

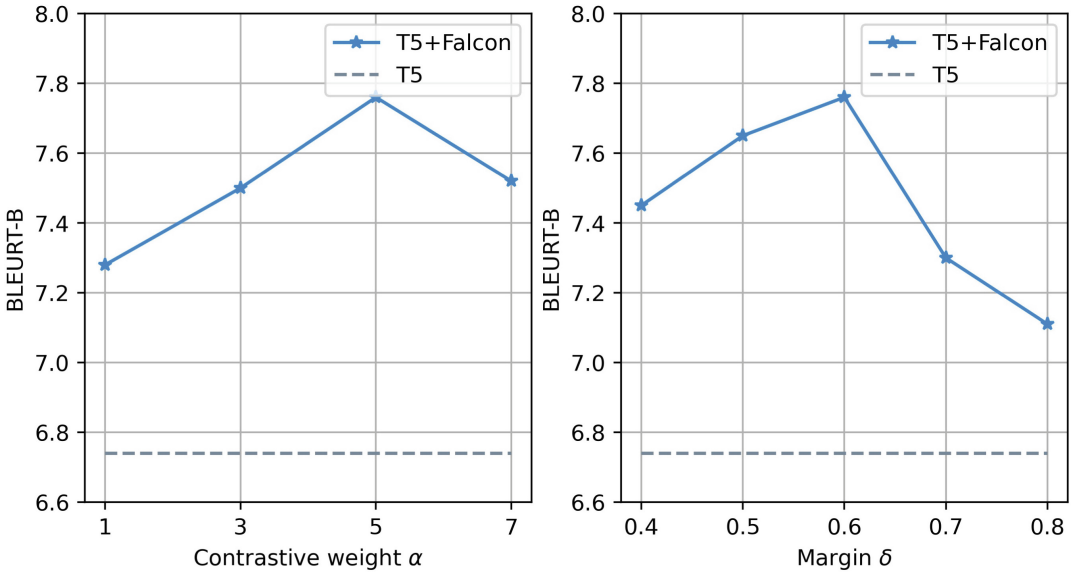

Fig. 3. Evaluation results with different contrastive weight α and margin δ using T5.

training. (a) and (e) in Table 4 demonstrates the impacts of contrastive training. It is observed that with contrastive training, the model improves the performance on all of the automatic metrics, including two new metrics to evaluate faithfulness. We also explore the effects of source imposters and target imposters on the performance of the model. As shown in Table 4 (c) and (d), training with source imposters or target imposters leads to a consistent performance improvement, which suggests that the model can benefit from perceiving factual differences during contrastive training.

Effect of Contrastive Inference. We propose contrastive inference to generate more faithful responses from contrasting source imposters. Comparing (a) and (b) demonstrates that using contrastive inference leads to a consistent performance increase. Comparing (e) and (f) also shows that contrastive inference contributes to improve the factual correctness of the generated responses.

Impact of Hyper-parameter α and δ. As shown in Fig. 3, we explore the impact of using different contrastive weight α and margin δ. We observe that as α or δ increase, the performance of T5 with FALCON presents an earlier increase and later decrease trend. The increase of the two hyper-parameters means that the importance of contrast learning boosts. We conjecture that a higher contrastive learning loss allows the model better distinguish imposters from ground truths, meanwhile it also risks failing to fit the original response.

6.4 Case Study

As shown in Fig. 4, we further demonstrate several typical examples for better understanding how our framework boosts faithfulness. Each example includes an input including SQL and table, a reference response (**Ref**), the T5 output (**T5**), and the generated response by our approach (**T5+FALCON**).

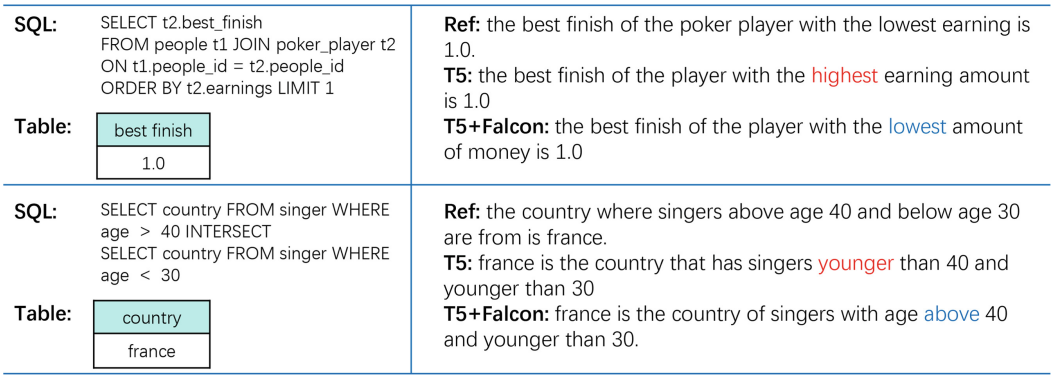

Fig. 4. Examples of generated responses from T5 and T5 with FALCON. The red color indicates factual errors in generated responses. Text in blue emphasizes the role of FALCON in improving factual correctness. (Color figure online)

In the first example, T5 fails to understand the condition that sort players by earning in ascending order and incorrectly generates a response with "*highest earning*". The reason can be that the "*highest*" occurs frequently in the training corpus. On the other hand, with the contrastive examples' constraint, FALCON guides T5 to generate a faithful response with the key fact "*lowest*".

The second example shows a similar situation, where T5 generates a grammatically correct response but with factual errors. "*above age 40*" is incorrectly translated as "*younger than 40*". The possible reason is that "*40*" is incorrectly associated with the operator < in the latter part of SQL. FALCON helps recover the correct meaning: *with age above 40*".

7 Conclusion

We propose a faithful contrastive generation framework FALCON to improve responses' factual correctness in a TableQA system. Our framework includes contrastive training and contrastive inference. To better evaluate faithfulness, we propose two metrics. Extensive evaluations show FALCON brings a favorable performance improvement amongst various baseline methods.

Acknowledgements. We thank anonymous reviewers for their comments and suggestions. This work was supported by National Key Research and Development Project (No. 2020AAA0109302), Shanghai Science and Technology Innovation Action Plan (No. 19511120400) and Shanghai Municipal Science and Technology Major Project (No. 2021SHZDZX0103).

References

1. Burges, C.J.C., et al.: Learning to rank using gradient descent. In: Proceedings of ICML. ACM International Conference Proceeding Series (2005)
2. Cai, D., Lam, W.: Graph transformer for graph-to-sequence learning. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020 (2020)

3. Cai, R., Liang, Z., Xu, B., Li, Z., Hao, Y., Chen, Y.: TAG : type auxiliary guiding for code comment generation. In: Proceedings of ACL (2020)
4. Cao, R., Chen, L., Chen, Z., Zhao, Y., Zhu, S., Yu, K.: LGESQL: line graph enhanced text-to-SQL model with mixed local and non-local relations. In: Proceedings of ACL (2021)
5. Cao, Z., Wei, F., Li, W., Li, S.: Faithful to the original: fact aware neural abstractive summarization. In: Proceedings of AAAI (2018)
6. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.E.: A simple framework for contrastive learning of visual representations. In: Proceedings of ICML. Proceedings of Machine Learning Research (2020)
7. Chen, W., Chen, J., Su, Y., Chen, Z., Wang, W.Y.: Logical natural language generation from open-domain tables. In: Proceedings of ACL (2020)
8. Cohen, J.: A coefficient of agreement for nominal scales. Educ. Psychol. Meas. **20**(1), 37–46 (1960)
9. Falke, T., Ribeiro, L.F.R., Utama, P.A., Dagan, I., Gurevych, I.: Ranking generated summaries by correctness: an interesting but challenging application for natural language inference. In: Proceedings of ACL (2019)
10. Gabriel, S., Celikyilmaz, A., Jha, R., Choi, Y., Gao, J.: GO FIGURE: a meta evaluation of factuality in summarization. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021 (2021)
11. Goodrich, B., Rao, V., Liu, P.J., Saleh, M.: Assessing the factual accuracy of generated text. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2019, 4–8 August 2019, Anchorage (2019)
12. Guo, J., et al.: Towards complex text-to-SQL in cross-domain database with intermediate representation. In: Proceedings of ACL (2019)
13. Hu, X., Li, G., Xia, X., Lo, D., Jin, Z.: Deep code comment generation. In: 2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC). IEEE (2018)
14. Iyer, S., Konstas, I., Cheung, A., Zettlemoyer, L.: Summarizing source code using a neural attention model. In: Proceedings of ACL (2016)
15. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of ICLR (2015)
16. Koncel-Kedziorski, R., Bekal, D., Luan, Y., Lapata, M., Hajishirzi, H.: Text generation from knowledge graphs with graph transformers. In: Proceedings of NAACL-HLT (2019)
17. Kryscinski, W., Keskar, N.S., McCann, B., Xiong, C., Socher, R.: Neural text summarization: a critical evaluation. In: Proceedings of EMNLP (2019)
18. Lee, S., Lee, D.B., Hwang, S.J.: Contrastive learning with adversarial perturbations for conditional text generation. In: Proceedings of ICLR (2021)
19. Lewis, M., et al.: BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proceedings of ACL (2020)
20. Lin, C.Y.: ROUGE: a package for automatic evaluation of summaries. In: Text Summarization Branches Out (2004)
21. Lin, X.V., Socher, R., Xiong, C.: Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In: Findings of the Association for Computational Linguistics: EMNLP 2020 (2020)
22. Liu, Y., et al.: RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint (2019)
23. Logeswaran, L., Lee, H.: An efficient framework for learning sentence representations. In: Proceedings of ICLR (2018)
24. Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint (2018)
25. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of ACL (2002)

26. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint (2019)
27. Scholak, T., Schucher, N., Bahdanau, D.: PICARD: parsing incrementally for constrained auto-regressive decoding from language models. arXiv preprint (2021)
28. Sellam, T., Das, D., Parikh, A.: BLEURT: learning robust metrics for text generation. In: Proceedings of ACL (2020)
29. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4–9 December 2017, Long Beach (2017)
30. Verma, V., Luong, T., Kawaguchi, K., Pham, H., Le, Q.V.: Towards domain-agnostic contrastive learning. In: Proceedings of ICML. Proceedings of Machine Learning Research (2021)
31. Wang, B., Shin, R., Liu, X., Polozov, O., Richardson, M.: RAT-SQL: relation-aware schema encoding and linking for text-to-SQL parsers. In: Proceedings of ACL (2020)
32. Wang, Z., Wang, X., An, B., Yu, D., Chen, C.: Towards faithful neural table-to-text generation with content-matching constraints. In: Proceedings of ACL (2020)
33. Xu, K., Wu, L., Wang, Z., Feng, Y., Sheinin, V.: SQL-to-text generation with graph-to-sequence model. In: Proceedings of EMNLP (2018)
34. Yang, Z., Cheng, Y., Liu, Y., Sun, M.: Reducing word omission errors in neural machine translation: a contrastive learning approach. In: Proceedings of ACL (2019)
35. Yu, T., et al.: CoSQL: a conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In: Proceedings of EMNLP (2019)
36. Yu, T., et al.: Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In: Proceedings of EMNLP (2018)
37. Zeng, J., et al.: Photon: a robust cross-domain text-to-SQL system. In: Proceedings of ACL (2020)
38. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: BERTScore: evaluating text generation with BERT. In: Proceedings of ICLR (2020)
39. Zhang, Y., Merck, D., Tsai, E., Manning, C.D., Langlotz, C.: Optimizing the factual correctness of a summary: a study of summarizing radiology reports. In: Proceedings of ACL (2020)
40. Zhong, V., Xiong, C., Socher, R.: Seq2sql: Generating structured queries from natural language using reinforcement learning. arXiv preprint (2017)
41. Zhong, W., et al.: LogicalFactChecker: leveraging logical operations for fact checking with graph module network. In: Proceedings of ACL (2020)