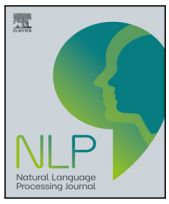
Natural Language Processing Journal 10 (2025) 100135

Contents lists available at ScienceDirect

Natural Language Processing Journal

journal homepage: www.elsevier.com/locate/nlp



Fine-tuning text-to-SQL models with reinforcement-learning training objectives

Xuan-Bang Nguyen [a,b,*], Xuan-Hieu Phan [a], Massimo Piccardi [c]

[a] University of Engineering and Technology, Vietnam National University, Hanoi, Viet Nam
[b] FPT Technology Research Institute, FPT University, Hanoi, Viet Nam
[c] Faculty of Engineering and Information Technology, University of Technology Sydney, Broadway, NSW, 2007, Australia

ARTICLE INFO

Keywords:
Text-to-SQL
Reinforcement learning
Reward functions
Policy-gradient algorithms
Fine-tuning

ABSTRACT

Text-to-SQL is an important natural language processing task that helps users automatically convert natural language queries into formal SQL code. While transformer-based models have pushed text-to-SQL to unprecedented accuracy levels in recent years, such performance is confined to models of very large size that can only be run in specialised clouds. For this reason, in this paper we explore the use of reinforcement learning to improve the performance of models of more conservative size, which can fit within standard user hardware. As reinforcement learning reward, we propose a novel function which better aligns with the text-to-SQL evaluation metrics, applied in conjunction with two strong policy gradient algorithms, REINFORCE and RELAX. Our experimental results over the popular Spider benchmark show that the proposed approach has been able to outperform a conventionally-trained T5 Small baseline by 6.6 pp (percentage points) of exact-set-match accuracy and 4.6 pp of execution accuracy, and a T5 Base baseline by 2.0 pp and 1.9 pp, respectively. The proposed model has also achieved a remarkable comparative performance against ChatGPT instances.

1. Introduction

Text-to-SQL is an important application of natural language processing (NLP) that converts database queries phrased in natural language into actual, executable SQL code. From their modest beginnings (Popescu et al., 2003), text-to-SQL models have made enormous strides in terms of performance, especially since the advent and wide adoption of transformers (Choi et al., 2021; Scholak et al., 2021; Li et al., 2023). However, the state-of-the-art accuracy is held by language models of very large size, which are perfectly suitable for cloud-based implementations, but which are raising increasing questions in terms of sustainability and flexibility of deployment. For instance, the expectation that language models may be deployed as "on-device" mobile implementations in the near future will substantially constrain their maximum size. An example of these models is the recently-released Gemini Nano, which specifically targets mobile devices (Team et al., 2023). Also, security and distributed requirements such as those entailed by federated learning (FL) impose drastic limitations on workable models' sizes. For these reasons, it is still very important to explore the potential of models of more parsimonious volume and investigate ways to improve their performance.

In the broader space of training approaches, reinforcement learning has repeatedly proved itself as an effective framework for training a model based on a trial-and-error concept (Sutton and Barto, 2018). Its success relies on the design of adequate "reward functions" that are able to incrementally guide the model to improve the accuracy of its predictions. This framework has led to many successful implementations in NLP, in particular in generative tasks such as machine translation and summarisation (Edunov et al., 2018a; Nguyen et al., 2017; Li et al., 2019; Parnell et al., 2022). Given that text-to-SQL, too, is essentially a generative task, in this paper we explore contemporary reinforcement learning approaches to improve the performance of small and base transformers at a complete parity of model size and training resources. The main contributions of our paper are:

- the use of reinforcement learning objectives to improve text-to-SQL performance compared to the standard negative log-likelihood (NLL) baseline;
- a reward function specialised for text-to-SQL predictions;
- experimental results over the challenging Spider benchmark dataset showing that the proposed approach has been able to outperform the NLL baseline and a popular policy gradient algorithm (PPO Schulman et al., 2017) in all cases, and perform strongly against ChatGPT instances.

* Corresponding author at: FPT Technology Research Institute, FPT University, Hanoi, Viet Nam.
E-mail addresses: bangnbx@fpt.com (X.-B. Nguyen), hieupx@vnu.edu.vn (X.-H. Phan), massimo.piccardi@uts.edu.au (M. Piccardi).

https://doi.org/10.1016/j.nlp.2025.100135
Received 23 July 2024; Received in revised form 30 January 2025; Accepted 17 February 2025

The rest of the paper is organised as follows: Section 2 reviews the related work. Section 3 introduces the reinforcement learning training objectives and presents the proposed approach. Section 4 describes the experimental set-up while Section 5 presents our results along with a detailed analysis. Finally, Section 6 recapitulates our work and concludes the paper.

2. Related work

The text-to-SQL task dates at least from the seminal work of Popescu et al. (2003). After many years of relatively slow progress, the field experienced a surge with the advent of deep learning, in particular with transformer-based models (Vaswani et al., 2017). Wang et al. (2020) have proposed RAT-SQL, a graph framework that uses a multiple relation-aware self-attention layers for schema encoding and linking, as well as a dedicated input representation. Lin et al. (2020) have proposed BRIDGE, an architecture that represents the utterance and database schema in the form of a tagged sequence. The sequence is then fed to a BERT encoder, and the SQL queries are generated by a pointer-generator network. RYANSQL (Choi et al., 2021) has defined the Statement Position Code (SPC) that decomposes any SQL query into a set of non-nested SELECT statements, which are then generated by a sketch-based slot filling approach. Scholak et al. (2021) have proposed a popular constrained-generation method called PICARD that generates effective SQL sequences using incremental parsing. This method is immediately applicable to any conventional transformer, without the need for altering the model training process. A large number of other works (Li et al., 2023; Hu et al., 2023; Zeng et al., 2023; Qi et al., 2022) have used PICARD as a component of their models to boost performance.

Recently, the rise of large language models (LLMs) has led to their application to the text-to-SQL task, too. However, the reported accuracies vary greatly. For instance, the works of Liu et al. (2023) and Dong et al. (2023) have showed that the execution accuracy of ChatGPT[1] can be impressive. With the help of sophisticated prompt engineering, they have achieved 70%–82% execution accuracy on the Spider validation split. Conversely, Ojha et al. (2024) have evaluated the performance of Llama-V2 7B (Touvron et al., 2023) in multiple text-to-SQL scenarios, reporting an accuracy on the Spider validation split of only 11.8% without any fine-tuning, compared to the 89.4% of a fine-tuned set-up. These studies are still in relatively early stages and major improvements should be expected in the near future. In the Spider leaderboard[2], the current top performer is a model called MiniSeek which has achieved 81.5% exact-set-match accuracy and 91.2% execution accuracy. In terms of exact-set-match, the second best work is the combination of Graphix-3B (Li et al., 2023) and PICARD (Scholak et al., 2021) which has achieved 74.0% accuracy. The next model is SHiP (Hu et al., 2023) which utilises an SQL synthesis framework and PICARD (Scholak et al., 2021) to boost performance to 73.1%. In terms of execution accuracy, the two models immediately below MiniSeek are based on GPT-4 variants (Gao et al., 2023). These two works have achieved 86.6% and 86.2% accuracy, respectively.

Spider is a large-scale text-to-SQL benchmark dataset released by Yu et al. (2018). It covers 138 domains with 200 databases of multiple tables, and contains over 10000 user inputs and 5600 SQL queries. To this day, Spider can be regarded as the most complex and popular dataset for single turn text-to-SQL generation. Other popular datasets include ATIS (Hemphill et al., 1990), GeoQuery (Zelle and Mooney, 1996), and WikiSQL (Zhong et al., 2017). However, ATIS and GeoQuery only cover a single database and domain, while WikiSQL only contains simple queries executed over single tables. In addition, Spider has maintained a leaderboard which has attracted a large amount of submissions and made comparisons easier. For these reasons, we have decided to carry out our experiments using Spider and its derivative variants. The training, validation, and test splits of Spider contain 8659, 1034, and 2147 samples, respectively (Yu et al., 2018).

[1] https://chat.openai.com

[2] https://yale-lily.github.io/spider

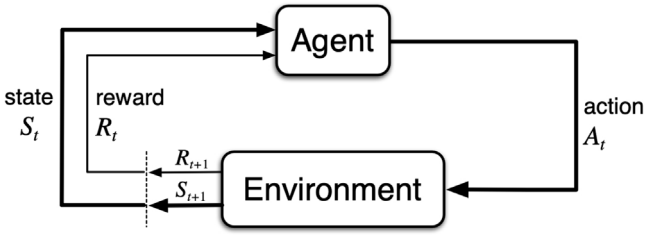


Fig. 1. Reinforcement Learning (Sutton and Barto, 2018).

At its turn, reinforcement learning is a popular machine learning paradigm that, in addition to many other areas, has found successful application in a number of natural language generation tasks, *e.g.* summarisation (Paulus et al., 2018; Pasunuru and Bansal, 2018; Parnell et al., 2022) and machine translation (Ranzato et al., 2016; Edunov et al., 2018b). However, to the best of our knowledge it has not been explored for text-to-SQL to this day, and for this reason its potential is the focus of our work. For the reader's benefit, we recap the key concepts of reinforcement learning in the following section.

3. Methodology

3.1. Reinforcement learning

The standard supervised solution for training a sequence-to-sequence model is to use maximum-likelihood estimation of its parameters, θ. To this aim, training minimises the following negative log-likelihood (NLL) loss:

$$L_{NLL} = -\log p_\theta(y|x)$$

where x represents an input sentence in the training set and y represents its ground-truth output sentence. While very popular, there are two inherent limitations with this training objective: (1) the model only learns to maximise the probability of the ground-truth references, y_i, without paying attention to any alternative labellings; and (2) minimising this loss function does not necessarily translate into achieving the best possible evaluation scores (on new data, but also on the training data themselves).

Reinforcement learning (RL) has been extensively employed to address these shortcomings. First, it allows a model to learn from imperfect output samples; for instance, the predictions of a partially-trained model. Along with the ground-truth references, using imperfect but assessable samples could be beneficial in terms of model's performance. Second, reinforcement learning is able to leverage evaluation metrics in the training process (such as BLEU for machine translation and exact-set-match accuracy for text-to-SQL generation), closing the gap between the training objective and the evaluation scores.

A classic reinforcement learning framework is shown in Fig. 1. At every step, the *agent*, which is in an identifiable *state*, S_t, takes an *action* A_t, receives *reward* R_t from the *environment*, and moves to a new state. Amongst the many problems that can be cast in this framework, we can even include the training of a sequence-to-sequence model: in this case, the agent is the model, which interacts with the environment (the training data), and its state are its parameters. At each step, the model takes an action by predicting an output sentence, receives a reward for it, and adjusts its parameters accordingly (the new state). What is left to do is design a reward function that can provide a useful score for the predicted output. In sequence-to-sequence training, the reward is typically provided only when the generation process reaches

an end-of-sentence token. In case a reward is sought at every step of the prediction, other techniques such as reward shaping (Ng et al., 1999) can be applied.

Reinforcement learning approaches, at their turn, subdivide into several styles (Sutton and Barto, 2018). For NLP tasks, *policy gradient* methods have been applied widely in recent years. These approaches do not attempt to directly optimise a target "value", but rather a "policy" (an action model) that can lead to it (Sutton and Barto, 2018). In this work, we have conducted experiments with two performing policy gradient methods, namely REINFORCE (Williams, 1992) and RELAX (Grathwohl et al., 2018), which we briefly recap in the following.

3.1.1. REINFORCE

Different from supervised learning, reinforcement learning encourages a model to generate predictions with high "reward", rather than just assigning high probability to some ideal references (the ground truth for a given input). In the case of text-to-SQL, the model's prediction is the SQL code for the given query, and a useful reward function could be any of the evaluation metrics described in Section 4.3. The goal of reinforcement learning is to maximise a training objective known as the *expected reward*:

$$RL = \mathbb{E}_{p_\theta(y|x)}[R(y)], \tag{1}$$

which is the expectation of the reward function for output y, $R(y)$, over the probability assigned by the model to that output, $p_\theta(y|x)$, given input x and model parameters θ. Maximising this objective encourages the model to assign high probability to outputs of high reward, directly improving its performance in the chosen evaluation metrics.

However, given that the reward is typically a discrete function of y, the RL objective is non-differentiable with respect to θ and cannot be maximised directly. A popular way to circumvent this issue is to ignore the dependency of R on θ and obtain this gradient based on the policy gradient theorem (Sutton and Barto, 2018):

$$\frac{\partial}{\partial\theta} RL = \mathbb{E}_{p_\theta(y|x)}[R(y)\frac{\partial}{\partial\theta}\log p_\theta(y|x)] \tag{2}$$

In turn, the well-known REINFORCE algorithm (Williams, 1992) approximates the expectation operator by sampling from $p_\theta(y|x)$:

$$\begin{aligned}&\mathbb{E}_{p_\theta(y|x)}[R(y)\frac{\partial}{\partial\theta}\log p_\theta(y|x)]\\ &\quad\approx \sum_{s=1}^{S} R(y)\frac{\partial}{\partial\theta}\log p_\theta(y^s|x), \quad y^s \sim p_\theta(y|x)\end{aligned} \tag{3}$$

Given that all contemporary programming libraries such as TensorFlow and PyTorch include auto-differentiation tools, we can then define the training loss function as:

$$L_{REINFORCE} = -\sum_{s=1}^{S} R(y)\log p_\theta(y^s|x), \quad y^s \sim p_\theta(y|x) \tag{4}$$

Note that we have added a minus sign to make the function a loss value to be minimised, and $R(y)$ is treated as a constant to be excluded from the differentiation chain.

3.1.2. RELAX

REINFORCE is referred to as an *unbiased* algorithm, meaning that its estimated gradient has the same expectation as the theoretical gradient. However, its drawback is having a *very high variance*, which directly derives from the straight sampling scheme and its impact on the gradient. Such a high variance can become a problem in practice, leading to trained instances of very different performance or even failure of training convergence. A number of algorithms have been proposed to mitigate the variance, but many of them introduce a bias in the gradient. Fortunately, Grathwohl et al. in Grathwohl et al. (2018) have managed to derive an algorithm – RELAX – that is both unbiased and low-variance, and has empirically outperformed REINFORCE in many evaluations (*e.g.*, Grathwohl et al. (2018), Parnell et al. (2022)). The RELAX loss function can be concisely expressed as:

$$\begin{aligned}L_{RELAX} = &\sum_{s=1}^{S} -[R(y^s) - c_\phi(\tilde{z})]\log p_\theta(y^s|x) + c_\phi(z) - c_\phi(\tilde{z}),\\ &y^s \sim p_\theta(y|x)\end{aligned} \tag{5}$$

where c_ϕ is a so-called *control variate* and z and $\tilde{z}$ are two auxiliary variables. The great advantage of introducing these additional terms is that term $c_\phi(\tilde{z})$ is proven to positively correlate with reward $R(y^s)$, ensuring a reduction of the gradient's variance, and offset $c_\phi(z) - c_\phi(\tilde{z})$ is able to completely remove the bias from the gradient. Beyond these brief notes, all details of RELAX can be found in Grathwohl et al. (2018).

3.1.3. Mixed training objective

Although reinforcement learning is effective, completely relying on it could lead to undesirable trajectories in parameter space. Similarly to other work (Wu et al., 2016, 2018), to stabilise the training process, we introduce a weight variable, λ, to create a mixed training objective:

$$L = (1-\lambda) * L_{NLL} + \lambda * L_{RL} \tag{6}$$

where L_{RL} is either $L_{REINFORCE}$ or L_{RELAX}.

3.2. Reward function

In text-to-SQL research, two evaluation metrics are the most common: the execution accuracy (EX) and the exact-set-match accuracy (EM). We provide a detailed explanation of these metrics in Section 4.3. Several other metrics have also been proposed, such as for instance the component matching accuracy (CM) that separately assesses the different components of the output (Yu et al., 2018). However, for a single SQL output, all these metrics can only have a value of either 0 or 1 (*i.e.*, completely wrong or perfect), and cannot therefore provide nuances on the quality of the model's predictions. We believe that SQL outputs which are not perfect, but have some valid components could still contribute to the learning process. For this reason, inspired by the component matching metric, we propose a reward function which can grade the individual prediction with intermediate scores between 0 and 1. Specifically, for every sample we determine how many components of the SQL query are correctly predicted out of all the components, and we express the reward as their ratio. The components we assess are: SELECT, WHERE, GROUP, ORDER, and IUEN. The meaning of the first four is immediate, while IUEN stands for the combination of the INTERSECT, UNION and EXCEPT operators. The full details of the proposed reward function are provided in Algorithm 1. As an example, consider the prediction and ground truth below:

```
Prediction: SELECT name FROM students WHERE
class = 'A1' ORDER BY grade;

Ground truth: SELECT name FROM students
WHERE id > 10 ORDER BY grade;
```

In the prediction, the SELECT and ORDER components are correct. Instead, the WHERE clause should be `WHERE id $>$~10` rather than `class = 'A1'`. In the computation of the reward, we also assume the components absent from the ground truth (*i.e.*, GROUP and IUEN in this case) to be notionally "correct", leading to a reward of 4/5 for this example.

Algorithm 1 Reward function.

Input: Sampled SQL y^s and ground truth SQL y^{tr}
Output: The reward score of y^s

```
count ← 0
for type in SELECT, WHERE, GROUP, ORDER, IUEN do
  if type in y^s and type in y^tr then
    if components of type in y^s = components of type in y^tr then
      count ← count + 1
    end if
  else if type not in y^s and type not in y^tr then
    count ← count + 1
  end if
end for
return count/5
```

Table 1
T5 models and their sizes (in 32-bit floating-point).

Model	Parameters	Size
Small	60M	242 MB
Base	220M	892 MB
Large	770M	2.95 GB
3B	3B	11.4 GB
11B	11B	45.2 GB

4. Experimental set-up

4.1. Dataset

As described in Section 2, we have chosen the Spider dataset for conducting the experiments because of its complexity compared to other available datasets. For this dataset, the ground truth of the test split has never been disclosed, and the test-set performance evaluation used to be carried out by the leaderboard maintainers. However, they have recently retired the test facility and performance evaluation over the test set is not anymore possible. Therefore, we have decided to fine-tune and validate our model using the training split alone, and report its performance over the validation split, as common in other work (*e.g.*, Zhang et al., 2024).

4.2. Model set-up

T5, introduced in Raffel et al. (2020), is an encoder–decoder model that can be used for all common NLP tasks by framing them into a text-to-text format. Prior to release, T5 has been pretrained on a very large amount of training data, and is meant to be used for downstream tasks in transfer-learning style. Similarly to other pretrained transformers, T5 comes in different sizes (Table 1). Of these sizes, the large, 3B, and 11B can be regarded as very imposing even by nowadays standards, requiring dedicated hardware, especially GPUs with large memories, to be deployed on. For these reasons, for the experiments we have focussed on T5 small and T5 base, which are more manageable for ordinary users and permit non-cloud deployment. The steps that we have followed for fine-tuning are:

1. fine-tune T5 for text-to-SQL with the standard NLL loss;
2. save two checkpoints (early and late);
3. fine-tune these checkpoints further with the proposed mixed training objective.

In the experiments, we have compared the models trained with the proposed mixed objective (for both REINFORCE and RELAX) with a model trained solely with the NLL and one trained with a popular policy gradient method, Proximal Policy Optimisation (PPO) (Schulman et al., 2017). In addition, in Section 5.3 we report a comparison with ChatGPT instances. All our experiments have fit within a single NVIDIA A100 40 GB card.

Table 2
Main results for the compared approaches over the Spider validation split. The dash marks are for cases where the validation accuracy degenerated to near zero.

Method		T5 small		T5 base	
		EM	EX	EM	EX
NLL		45.8	46.2	57.4	60.2
PPO		44.9	44.2	54.1	55.3
REINFORCE	$\lambda = 0.05$	47.2	47.1	**58.6**	60.9
	$\lambda = 0.1$	50.4	49.5	57.6	60.1
	$\lambda = 0.2$	51.0	49.6	57.9	**61.6**
	$\lambda = 0.3$	51.3	50.1	–	–
	$\lambda = 0.5$	**52.4**	**50.8**	–	–
RELAX	$\lambda = 0.1$	50.3	50.6	**59.4**	**62.1**
	$\lambda = 0.3$	**50.8**	**50.7**	59.3	61.1
	$\lambda = 0.5$	50.4	**50.7**	58.6	60.0

4.3. Evaluation metrics

Two common metrics for text-to-SQL (and the Spider leaderboard in particular) are the exact-set-match accuracy and the execution accuracy:

Exact-Set-Match Accuracy (EM) A mere string-to-string comparison between prediction and ground truth is not adequate for SQL, because the ordering of the SQL clauses is irrelevant. For example, these two SQL queries are equivalent:

```
SELECT name FROM students WHERE
name LIKE 'John%' AND class_id = 1;

SELECT name FROM students WHERE
class_id = 1 AND name LIKE 'John%';
```

To address this problem, exact-set matching organises the clauses' components as sets, and then compares prediction and ground truth set-by-set to discount the components' order.

Execution Accuracy (EX) For a single input, there might be multiple SQL queries that are correct in terms of logic and produce the same result. As such, the Execution Accuracy assesses the accuracy of the output after performing the SQL query on the corresponding database.

5. Results

Table 2 presents the results for the two T5 models. For clarity of presentation, we discuss them separately in the following.

5.1. T5 small

The leftmost columns of Table 2 show the EM and EX accuracies for the T5 small models. The results show that REINFORCE has been able to improve the EM metric by up to 6.6 percentage points (pp) and the EX metric by 4.6 pp compared to the NLL-trained model. At its turn, RELAX has achieved a comparable improvement of 4.5 pp of EX, but only 5.0 pp of EM. The results also show that PPO's performance has actually been worse than that of the NLL, despite the fact that we have spent effort trying to optimise all its key parameters (the Kullback–Leibler target, the batch size, the mini-batch size, and the clipping range). However, we cannot exclude that a different parametrisation could lead to better results.

As the two checkpoints for fine-tuning with the mixed training objective we have employed: (1) the checkpoint of NLL fine-tuning where the validation accuracy had become steady (step 768); (2)

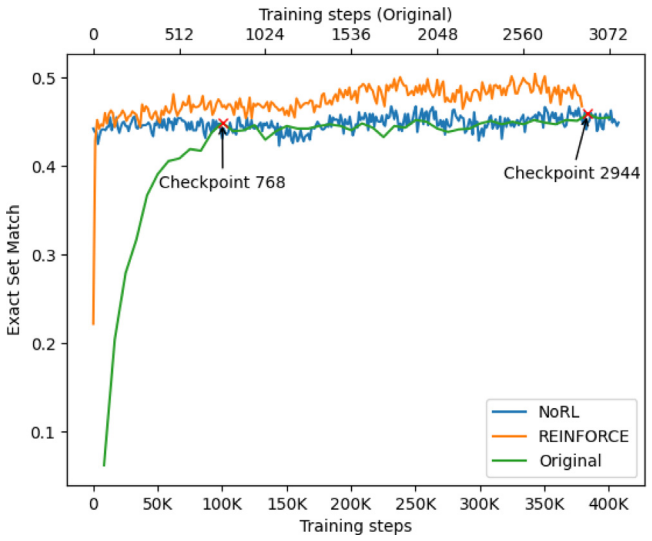


Fig. 2. EM accuracy along the training steps. *Original* refers to the standard NLL fine-tuning (it has separated x-axis for better visualisation); *NoRL* to a model fine-tuned using the NLL, but started from the checkpoint at step 768 for parity with REINFORCE; *REINFORCE* to a model fine-tuned with the proposed objective, also started from the same checkpoint.

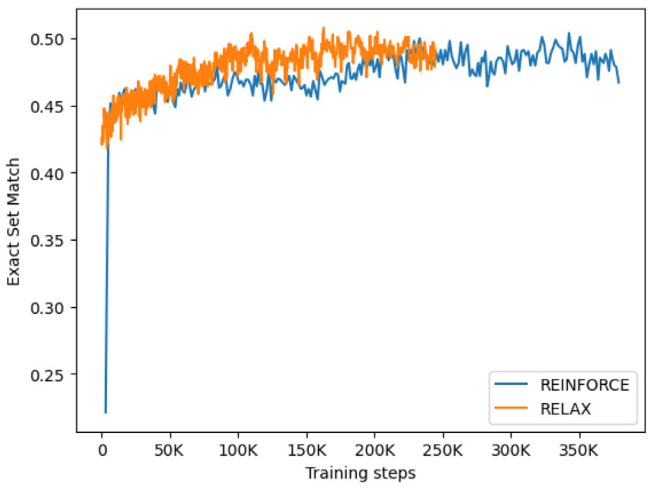


Fig. 3. Comparison of REINFORCE and RELAX along the training steps. Note that the RELAX training terminated after about 250 K steps when it reached convergence.

the checkpoint of NLL fine-tuning where the validation accuracy had reached its peak (step 2944). These two steps were annotated in Fig. 2. Our initial experiments have showed that the first, early checkpoint is more effective for the ensuing reinforcement learning, and therefore is the one that we report. This behaviour might stem from the fact that the training-set predictions of this checkpoint are still imperfect, and provide a more useful reward signal for the reinforcement learning fine-tuning. Conversely, the training-set predictions of the later checkpoint are virtually perfect, and may render the reinforcement learning objective just a duplicate of the NLL. With a different wording, the fully-trained model is more likely to find itself in a local minimum of the reinforcement learning objective, and therefore be more difficult to improve. To provide more insights on the models' behaviour, Fig. 2 shows the plots of the validation accuracy along the training steps for the NLL model, the REINFORCE model ($\lambda = 0.1$) and an NLL model restarted from checkpoint 768 ("NoRL"), for complete parity of initial conditions with the REINFORCE model. The plots show that fine-tuning with the NLL again has not led to any noticeable gain in performance. Conversely, REINFORCE has been able to improve the accuracy by almost five percentage points (pp). In turn, Fig. 3 shows the comparative performance of REINFORCE and RELAX. The main difference between the two approaches is that RELAX has been able to reach a comparable accuracy to REINFORCE in less than half the training steps.

5.2. T5 base

Similarly to T5 small, for T5 base we have chosen the early checkpoint at step 448 to conduct further fine-tuning. The two rightmost columns of Table 2 show that, for this model, using REINFORCE has led to a gain of 1.2 EM pp and 1.4 EX pp compared to the NLL. For this model, RELAX has achieved a larger improvement than REINFORCE, of 2.0 EM pp and 1.9 EM pp compared to the NLL. Overall, the improvements for this model have been less than for T5 small, but this was somehow to be expected given the much higher levels of accuracy of the NLL baseline. Once again, the further fine-tuning with PPO has led to a lower accuracy than the NLL baseline.

Table 2 also shows that for a relatively large value of λ (0.3), the REINFORCE model's accuracy has collapsed to zero. We ascribe this behaviour to an excessive amplification of the policy gradient that might have led to a degenerate parametrisation. Conversely, in the lower range (0.01 – 0.2), the performance has remained rather stable.

A behaviour that is worth noting is that the improvements of EM are larger than those of EX. The likely reason is that our reward function more closely aligns with the EM than the EX, and eventually drives the model to generate output with higher EM values on average.

5.3. ChatGPT

Pretrained large language models (LLMs) such as ChatGPT[3] are capable of performing an unprecedented variety of NLP tasks, not least text-to-SQL. For this reason, we have carried out an additional experiment using GPT-3.5 Turbo and GPT-4 with zero- and one-shot set-ups.

For the zero-shot set-up, we have used a simple prompt directly constructed from the DB schema. Below is an example that was fed into the ChatGPT API:

```
### Complete sqlite SQL query only and
with no explanation
### Sqlite SQL tables, with their properties:
#
# STADIUM(stadium_id,location,name,capacity,
highest,lowest,average);
# SINGER(singer_id,name,country,song_name,
song_release_year,age,is_male);
#
### How many singers do we have?
SELECT
```

For the one-shot set-up, we have preceded the user query with an example of conversion. In the *fixed* set-up, we have implemented this by manually selecting a "rich" example, and using it for all samples. Since the choice is arbitrary, we simple chose the first sample of the training set. In our qualitative judgment, it is an example of good and comprehensive quality. In the *random* set-up, it has been done by randomly selecting an example for each sample. An example of this prompt is:

[3] https://chat.openai.com

Table 3
GPT models performance on Spider; *fixed* and *random* denote how the example was chosen: *fixed* used the same example for every query, while *random* used a random example per query.

Method	GPT model	EM	EX
Zero-shot	3.5 Turbo	48	60.8
	4	44.2	54.5
One-shot fixed	3.5 Turbo	**48.9**	**63.4**
	4	44.8	56.8
One-shot random	3.5 Turbo	46.1	59.6
	4	46.4	56.8

```
### Complete sqlite SQL query only and
with no explanation
### First, I will give you an example
### Example sqlite SQL tables,
 with their properties:
#
# HIGHSCHOOLER(id,name,grade);
# FRIEND(student_id,friend_id);
# LIKES(student_id,liked_id);
#
### Example question: Show the names of
high schoolers who have at least 3 friends.
### Example result: select t2.name from friend
as t1 join highschooler as t2
 on t1.student_id = t2.id
group by t1.student_id having count(*) >= 3
#
### Below is your tables and question:
### Sqlite SQL tables, with their properties:
#
#STADIUM(stadium_id,location,name,capacity,
highest,lowest,average);
# SINGER(singer_id,name,country,song_name,
song_release_year,age,is_male);
# CONCERT(concert_id,concert_name,theme,
stadium_id,year);
# SINGER_IN_CONCERT(concert_id,singer_id);
#
### How many singers do we have?
SELECT
```

Table 3 reports the results from this experiment. The first notable finding is that the EX accuracy has been much higher than the EM accuracy, which we believe is due to a much larger amount and variety of SQL data in the LLM pretraining compared to the Spider's training set. A second comment is that the EM accuracy has been comparable or lower than those achieved with the T5 small model. Only the EX accuracy of the 3.5 Turbo model has been able to outperform those of the T5 models. However, to properly position these results, it has to be remarked that the performance of all LLMs can be substantially improved with a range of other techniques which include larger numbers of examples, chain-of-thought guidance, and retrieval-augmented generation (Qiao et al., 2023). Nevertheless, the difference in size between the models is staggering (175B parameters for GPT-3.5 and a rumored 1T parameters for GPT-4 vs 60M and 220M for T5 small and base, respectively). This further emphasises the benefits that can be achieved from an effective fine-tuning of much smaller models.

5.4. Dataset variants

To further probe our overall best model (*i.e.*, RELAX), we have tested it over three variants of the Spider dataset, namely: *Spider-DK*, *Spider-Realistic*, and *Spider-Syn*. The first dataset, Spider-DK (Gan et al., 2021b), introduces domain knowledge into a selection of Spider queries in order to reflect real-world questions. The second one, Spider-Realistic (Deng et al., 2021), removes the column names from the Spider validation set to create a more realistic and challenging test. Finally, Spider-Syn (Gan et al., 2021a) replaces words in the Spider's database schema with manually-selected synonyms. In real-world scenarios, user utterances are not in perfect forms but come with synonyms, inadequate schema information, or contains domain context. The fact that a model can keep its performance on those scenarios means that the model is well generalised and suitable to be used in practical environment.

Table 4
Results for RELAX on the Spider variants.

Dataset	NLL		RELAX	
	EM	EX	EM	EX
Spider	57.4	60.2	**59.4**	**62.1**
Spider-DK	31.2	38.7	**32.7**	**40.9**
Spider-Realistic	**50.2**	51.0	49.8	**51.2**
Spider-Syn	41.0	44.4	**45.6**	**48.5**

Table 4 shows the results for the T5 base RELAX model. In most cases, its EM and EX accuracies have been higher than those of the NLL, with the only exception of Spider-Realistic, where the NLL has reported a slightly higher EM than RELAX (50.2 vs. 49.8). RELAX' gain has been the largest on Spider-Syn, with more than 4 pp for both the EM and EX. These results show that RELAX has been able to lead to improvements not only on the original dataset, but also on variants with more challenging natural language queries.

6. Conclusions

In this paper, we have proposed an approach for fine-tuning text-to-SQL models with reinforcement-learning training objectives so as to boost their performance compared to the conventional NLL. To this aim, we have used two policy gradient approaches, namely REINFORCE and the more recent RELAX, which enjoys nice properties of unbiasedness and low variance. In addition, we have proposed a dedicated reward function that aligns with the evaluation metrics of text-to-SQL. In our experiments, we have focused on models of relatively small size (small and base T5 models) that can fit within standard GPU hardware and ensure maximum applicability. Our experimental results have shown that the proposed approach has been able to outperform the NLL baseline and a popular policy-gradient approach, PPO, over the Spider benchmark and a number of enhanced variants. In addition, the proposed approach has reported higher exact-set-match (EM) scores than ChatGPT instances even with the T5 small model, which is less than a thousand times smaller. In the future, we plan to investigate reward functions that can emphasise other aspects of text-to-SQL performance, and also explore models with larger numbers of parameters, yet in low-rank configurations that can still fit within standard memory constraints.

CRediT authorship contribution statement

Xuan-Bang Nguyen: Writing – review & editing, Writing – original draft, Visualization, Software, Project administration, Methodology. **Xuan-Hieu Phan:** Writing – review & editing, Supervision. **Massimo Piccardi:** Writing – review & editing, Supervision, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table A.5
Example where the NLL model has failed to join the required table car_makers.

Question	What is the name of the country with the most car makers?
Schema	continents: contid, continent countries: countryid, countryname, continent car_makers: id, maker, fullname, country model_list: modelid, maker, model car_names: makeid, model, make cars_data: id, mpg, cylinders, edispl, horsepower, weight, accelerate, year
Ground truth	select t2.countryname from car_makers as t1 join countries as t2 on t1.country = t2.countryid group by t1.country order by count(*) desc limit 1
NLL	select countryname from countries group by countryname order by count(*) desc limit 1
RELAX	select t1.countryname from countries as t1 join car_makers as t2 on t1.countryid = t2.country group by t1.countryid order by count(*) desc limit 1

Table A.6
Example where the NLL model has overcomplicated the output query, involving the unnecessary table airports.

Question	What are airlines that have flights arriving at airport 'AHD'?
Schema	airlines: uid, airline, abbreviation, country airports: city, airportcode (AHD), airportname, country, countryabbrev flights: airline, flightno, sourceairport (AHD), destairport (AHD)
Ground truth	select t1.airline from airlines as t1 join flights as t2 on t1.uid = t2.airline where t2.destairport = "AHD"
NLL	select t3.address from flights as t1 join airports as t2 on t1.airport = t2.airportcode join airlines as t3 on t1.airportno = t3.uid where t2.sourceairport = 'AHD'
RELAX	select t1.airline from airlines as t1 join flights as t2 on t1.uid = t2.airline where t2.sourceairport = 'AHD'

Appendix. Sample outputs of NLL and RL models

In this section, we provide a few examples of output from NLL- and RELAX-trained models to offer a detailed view of how reinforcement learning can improve text-to-SQL generation. In these tables, database schemas are shown in the following format:

```
table_1: column_1, column_2
table_2: column_1, column_2(column_content),
column_3
```

Table A.5 is an example where the NLL-trained model has failed to join the required table *car_makers*. The question asked for the country with the most car makers, and it is therefore mandatory to have table *car_makers* in the SQL query. Instead, the output from the NLL model only includes the name of the country that appears the most often, and does not satisfy the request.

The example in Table A.6 shows a case where the NLL model has output an unnecessarily complex SQL query, involving a table, *airports*, that should not be included.

In turn, Table A.7 shows an example where the NLL model has been confused by a sophisticated schema. The question is rather simple in its own right, and the expected query is a straightforward SELECT statement. However, the NLL model has joined an unrelated table and produced incorrect SQL code.

Finally, Table A.8 shows a case in which the NLL model has misinterpreted the database schema. Column *name* should not be used directly as a member of table *countrylanguage*; rather, it should be referred to by joining table *country* as in the ground-truth query.

As shown in the tables, the RELAX model has been able to mitigate all these types of mistakes.

Table A.7
Example in which the NLL model has been confused by a sophisticated schema and has failed to produce a simple query.

Question	What is the first and second line for all addresses?
Schema	addresses: address_id, line_1, line_2, line_3, city, zip_postcode, state_province_county, country, other_address_details, courses: course_id, course_name, course_description, other_details departments: department_id, department_name, department_description, other_details degree_programs: degree_program_id, department_id, degree_summary_name,degree_summary_description, other_details sections: section_id, course_id, section_name, section_description, other_details semesters: semester_id, semester_name, semester_description, other_details students: student_id, current_address_id, permanent_address_id, first_name, middle_name, last_name,cell_mobile_number, email_address, ssn, date_first_registered, date_left,other_student_details student_enrolment: student_enrolment_id, degree_program_id, semester_id, student_id, other_details student_enrolment_courses: student_course_id, course_id, student_enrolment_id transcripts: transcript_id, transcript_date, other_details transcript_contents: student_course_id, transcript_id
Ground truth	select line_1, line_2 from addresses
NLL	select t1.line_1, t1.line_2 from addresses as t1 join departments as t2 on t1.address_id = t2.department_id
RELAX	select line_1, line_2 from addresses

Table A.8
Example where the NLL model has misinterpreted the schema. Column name does not exist in table countrylanguage.

Question	What is the total number of languages used in Aruba?
Schema	city: id, name, countrycode, district, population country: code, name (Aruba), continent, region, surfacearea, indepyear, population, lifeexpectancy, gnp, gnpold, localname (Aruba), governmentform, headofstate, capital, code2 countrylanguage: countrycode, language, isofficial, percentage
Ground truth	select count(t2.language) from country as t1 join countrylanguage as t2 on t1.code = t2.countrycode where t1.name = "Aruba"
NLL	select count(distinct language) from countrylanguage where name = 'Aruba'
RELAX	select count(distinct language) from country as t1 join countrylanguage as t2 on t1.code = t2.countrycode where t1.name = "Aruba"

References

Choi, D., Shin, M.C., Kim, E., Shin, D.R., 2021. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. Comput. Linguist. 47 (2), 309–332.

Deng, X., Awadallah, A.H., Meek, C., Polozov, O., Sun, H., Richardson, M., 2021. Structure-grounded pretraining for text-to-SQL. In: Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., Zhou, Y. (Eds.), Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, pp. 1337–1350. http://dx.doi.org/10.18653/v1/2021.naacl-main.105, Online. URL: https://aclanthology.org/2021.naacl-main.105/.

Dong, X., Zhang, C., Ge, Y., Mao, Y., Gao, Y., Lin, J., Lou, D., et al., 2023. C3: Zero-shot text-to-sql with chatgpt. arXiv preprint arXiv:2307.07306.

Edunov, S., Ott, M., Auli, M., Grangier, D., Ranzato, M., 2018a. Classical structured prediction losses for sequence to sequence learning. In: Walker, M., Ji, H., Stent, A. (Eds.), Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). Association for Computational Linguistics, New Orleans, Louisiana, pp. 355–364. http://dx.doi.org/10.18653/v1/N18-1033, URL: https://aclanthology.org/N18-1033/.

Edunov, S., Ott, M., Auli, M., Grangier, D., Ranzato, M., 2018b. Classical structured prediction losses for sequence to sequence learning. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). Association for Computational Linguistics, New Orleans, Louisiana, pp. 355–364. http://dx.doi.org/10.18653/v1/N18-1033, URL: https://www.aclweb.org/anthology/N18-1033.

Gan, Y., Chen, X., Huang, Q., Purver, M., Woodward, J.R., Xie, J., Huang, P., 2021a. Towards robustness of text-to-SQL models against synonym substitution. In: Zong, C., Xia, F., Li, W., Navigli, R. (Eds.), Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Association for Computational Linguistics, pp. 2505–2515. http://dx.doi.org/10.18653/v1/2021.acl-long.195, Online. URL: https://aclanthology.org/2021.acl-long.195/.

Gan, Y., Chen, X., Purver, M., 2021b. Exploring underexplored limitations of cross-domain text-to-SQL generalization. In: Moens, M.-F., Huang, X., Specia, L., Yih, S.W.-t. (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, pp. 8926–8931. http://dx.doi.org/10.18653/v1/2021.emnlp-main.702, URL: https://aclanthology.org/2021.emnlp-main.702/.

Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., Zhou, J., 2023. Text-to-sql empowered by large language models: A benchmark evaluation. arXiv preprint arXiv:2308.15363.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., Duvenaud, D., 2018. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In: International Conference on Learning Representations.

Hemphill, C.T., Godfrey, J.J., Doddington, G.R., 1990. The ATIS spoken language systems pilot corpus. In: Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27,1990. URL: https://aclanthology.org/H90-1021.

Hu, Y., Zhao, Y., Jiang, J., Lan, W., Zhu, H., Chauhan, A., Li, A.H., Pan, L., Wang, J., Hang, C.-W., Zhang, S., Guo, J., Dong, M., Lilien, J., Ng, P., Wang, Z., Castelli, V., Xiang, B., 2023. Importance of synthesizing high-quality data for text-to-SQL parsing. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (Eds.), Findings of the Association for Computational Linguistics: ACL 2023. Association for Computational Linguistics, Toronto, Canada, pp. 1327–1343. http://dx.doi.org/10.18653/v1/2023.findings-acl.86, URL: https://aclanthology.org/2023.findings-acl.86/.

Li, J., Hui, B., Cheng, R., Qin, B., Ma, C., Huo, N., Huang, F., Du, W., Si, L., Li, Y., 2023. Graphix-T5: mixing pre-trained transformers with graph-aware layers for text-to-SQL parsing. In: Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence. AAAI'23/IAAI'23/EAAI'23, AAAI Press, http://dx.doi.org/10.1609/aaai.v37i11.26536.

Li, S., Lei, D., Qin, P., Wang, W.Y., 2019. Deep reinforcement learning with distributional semantic rewards for abstractive summarization. In: Inui, K., Jiang, J., Ng, V., Wan, X. (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing. EMNLP-IJCNLP, Association for Computational Linguistics, Hong Kong, China, pp. 6038–6044. http://dx.doi.org/10.18653/v1/D19-1623, URL: https://aclanthology.org/D19-1623/.

Lin, X.V., Socher, R., Xiong, C., 2020. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In: Cohn, T., He, Y., Liu, Y. (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2020. Association for Computational Linguistics, pp. 4870–4888. http://dx.doi.org/10.18653/v1/2020.findings-emnlp.438, Online. URL: https://aclanthology.org/2020.findings-emnlp.438/.

Liu, A., Hu, X., Wen, L., Yu, P.S., 2023. A comprehensive evaluation of ChatGPT's zero-shot text-to-sql capability. arXiv preprint arXiv:2303.13547.

Ng, A.Y., Harada, D., Russell, S., 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In: ICML, vol. 99, Citeseer, pp. 278–287.

Nguyen, K., Daumé III, H., Boyd-Graber, J., 2017. Reinforcement learning for bandit neural machine translation with simulated human feedback. In: Palmer, M., Hwa, R., Riedel, S. (Eds.), Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Copenhagen, Denmark, pp. 1464–1474. http://dx.doi.org/10.18653/v1/D17-1153, URL: https://aclanthology.org/D17-1153/.

Ojha, P.K., Gautam, A., Agrahari, A., Singh, P., 2024. SFT for improved text-to-SQL translation. Int. J. Intell. Syst. Appl. Eng. 12 (17s), 700–705.

Parnell, J., Jauregi Unanue, I., Piccardi, M., 2022. A multi-document coverage reward for RELAXed multi-document summarization. In: Muresan, S., Nakov, P., Villavicencio, A. (Eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 5112–5128.

Pasunuru, R., Bansal, M., 2018. Multi-reward reinforced summarization with saliency and entailment. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). Association for Computational Linguistics, New Orleans, Louisiana, pp. 646–653. http://dx.doi.org/10.18653/v1/N18-2102, URL: https://www.aclweb.org/anthology/N18-2102.

Paulus, R., Xiong, C., Socher, R., 2018. A deep reinforced model for abstractive summarization. In: International Conference on Learning Representations. URL: https://openreview.net/forum?id=HkAClQgA-.

Popescu, A.-M., Etzioni, O., Kautz, H., 2003. Towards a theory of natural language interfaces to databases. In: Proceedings of the 8th International Conference on Intelligent User Interfaces. pp. 149–157.

Qi, J., Tang, J., He, Z., Wan, X., Cheng, Y., Zhou, C., Wang, X., Zhang, Q., Lin, Z., 2022. RASAT: Integrating relational structures into pretrained Seq2Seq model for text-to-SQL. In: Goldberg, Y., Kozareva, Z., Zhang, Y. (Eds.), Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, pp. 3215–3229. http://dx.doi.org/10.18653/v1/2022.emnlp-main.211, URL: https://aclanthology.org/2022.emnlp-main.211/.

Qiao, S., Ou, Y., Zhang, N., Chen, X., Yao, Y., Deng, S., Tan, C., Huang, F., Chen, H., 2023. Reasoning with language model prompting: A survey. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (Eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Toronto, Canada, pp. 5368–5393. http://dx.doi.org/10.18653/v1/2023.acl-long.294, URL: https://aclanthology.org/2023.acl-long.294.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., 2020. Exploring the limits of transfer learning with a unified Text-to-Text transformer. J. Mach. Learn. Res. 21 (140), 1–67, URL: http://jmlr.org/papers/v21/20-074.html.

Ranzato, M., Chopra, S., Auli, M., Zaremba, W., 2016. Sequence level training with recurrent neural networks. In: Bengio, Y., LeCun, Y. (Eds.), 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings. URL: http://arxiv.org/abs/1511.06732.

Scholak, T., Schucher, N., Bahdanau, D., 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In: Moens, M.-F., Huang, X., Specia, L., Yih, S.W.-t. (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, pp. 9895–9901. http://dx.doi.org/10.18653/v1/2021.emnlp-main.779, URL: https://aclanthology.org/2021.emnlp-main.779/.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. MIT Press.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A.M., Hauth, A., et al., 2023. Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al., 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30.

Wang, B., Shin, R., Liu, X., Polozov, O., Richardson, M., 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, pp. 7567–7578. http://dx.doi.org/10.18653/v1/2020.acl-main.677, Online. URL: https://aclanthology.org/2020.acl-main.677/.

Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. 8, 229–256.

Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al., 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

Wu, L., Tian, F., Qin, T., Lai, J., Liu, T.-Y., 2018. A study of reinforcement learning for neural machine translation. In: Riloff, E., Chiang, D., Hockenmaier, J., Tsujii, J. (Eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Brussels, Belgium, pp. 3612–3621. http://dx.doi.org/10.18653/v1/D18-1397, URL: https://aclanthology.org/D18-1397/.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., Radev, D., 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In: Riloff, E., Chiang, D., Hockenmaier, J., Tsujii, J. (Eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. EMNLP 2018, Association for Computational Linguistics, pp. 3911–3921.

Zelle, J.M., Mooney, R.J., 1996. Learning to parse database queries using inductive logic programming. In: Proceedings of the National Conference on Artificial Intelligence. pp. 1050–1055.

Zeng, L., Parthasarathi, S.H.K., Hakkani-Tur, D., 2023. N-best hypotheses reranking for text-to-sql systems. In: 2022 IEEE Spoken Language Technology Workshop. SLT, IEEE, pp. 663–670.

Zhang, B., Ye, Y., Du, G., Hu, X., Li, Z., Yang, S., Liu, C.H., Zhao, R., Li, Z., Mao, H., 2024. Benchmarking the Text-to-SQL capability of large language models: A comprehensive evaluation. arXiv:2403.02951.

Zhong, V., Xiong, C., Socher, R., 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. CoRR. arXiv:1709.00103.