Natural Language Response Generation from SQL with Generalization and Back-translation

Saptarashmi Bandyopadhyay
University of Maryland, College Park
College Park, MD 20742
sapta.band59@gmail.com

Tianyang Zhao
The Pennsylvania State University
University Park, PA 16802
zty12713@gmail.com

Abstract

Generation of natural language responses to the queries of structured language like SQL is very challenging as it requires generalization to new domains and the ability to answer ambiguous queries among other issues. We have participated in the CoSQL shared task organized in the IntEx-SemPar workshop at EMNLP 2020. We have trained a number of Neural Machine Translation (NMT) models to efficiently generate the natural language responses from SQL. Our shuffled back-translation model has led to a BLEU score of 7.47 on the unknown test dataset. In this paper, we will discuss our methodologies to approach the problem and future directions to improve the quality of the generated natural language responses.

1 Introduction

Natural language interfaces to databases (NLIDB) has been the focus of many research works, including a shared track on the Conversational text-to-SQL Challenge at EMNLP-IntexSemPar 2020 (Yu et al., 2019). We have focused on the second task, natural language response generation from SQL queries and execution results.

For example, when the SQL query "SELECT dorm_name FROM dorm" is present, a possible response by the system could be "This is the list of the names of all the dorms". The ideal responses should demonstrate the results of the query, present the logical relationship between the query objects and the results, and be free from any grammatical error. Another challenge for this task is that the system needs to be able to generalize and do well on the SQL queries and the database schema which it has never seen before.

2 Related Works

Many existing papers focus on text to SQL generation like Shin (2019) and Zhong et al. (2017) which emphasize self-attention and reinforcement-learning-based approaches. The problem of generating natural language responses from SQL is that this specific area is relatively under-researched, but we have tried to come up with probable solutions in this shared task.

Gray et al. (1997) inspired us to generalize SQL keywords for better response generation with improvement in generalization. We have employed back-translation, used by Sennrich et al. (2015) and Hoang et al. (2018), in order to increase the BLEU score. We were also motivated by the linguistic generalization results pointed out by Bandyopadhyay (2019) and Bandyopadhyay (2020) where the lemma and the Part-of-Speech tag are added to the natural language dataset for better generalization. Although we did not include it in our final model due to challenges in removing the linguistic factors, this approach offers a potential future in the generalization of the generated natural language responses.

3 Pre-processing Methods

We decided to take the Neural Machine Translation (NMT) approach, where the SQL queries with the execution results are regarded as the source, and the natural language, more specifically English, responses are seen as the target. We chose Seq2seq as our baseline model. After several attempts of training and parameter tuning, we were able to obtain a baseline BLEU score.

In order to further improve the BLEU score, first, we came up with the idea of SQL keyword generalization. SQL keyword generalization is a pre-processing method we applied to the input data (i.e. the SQL queries with the execution results). We

Proceedings of the First Workshop on Interactive and Executable Semantic Parsing, pages 46–49

https://doi.org/10.18653/v1/P17

Original Keywords	Generalization
UNION, INTERSECTION, EXCEPT	SET
AND, OR	LOGIC
EXISTS, UNIQUE, IN	NEST
ANY, ALL	RANGE
AVG, COUNT, SUM, MAX, MIN	AGG

Table 1: The grouped SQL keywords and their substitutions.

first put the common SQL keywords into different groups based on their characteristics. Table 1 shows our choices of grouping. Then, we substituted each of those keywords in the input data to the newly purposed, generalized name according to the group we put the keyword in.

More specifically, UNION, INTERSECTION, and EXCEPT are substituted as SET because these three keywords are set operations. AND and OR are substituted as LOGIC because they are logic operators. One thing worth noting is that although AND in SQL is not only a logic operator as it can also be used to join tables, the phrase "JOIN ... ON ..." is primarily used for this particular purpose. EXISTS, UNIQUE, and IN are substituted as NEST because these keywords are followed by one or multiple nested queries. ANY and ALL are substituted as RANGE since they are followed by a sub-query that will return a range of values, and an operator such as $>$ is usually in front of ANY and ALL to compare with those values returned by the sub-query. AVG, COUNT, SUM, MAX, and MIN are substituted as AGG since all these keywords are aggregate operators.

The remaining common SQL keywords are difficult to be grouped with other ones. For example, GROUP BY and HAVING have distinct meanings and work differently as they are followed by non-identical elements. GROUP BY is followed by a "grouping-list", usually an attribute of a table, while HAVING is followed by a "group-qualification", usually a comparison involving an operator. Therefore, those keywords are kept as they are in the input data. Moreover, the operators are also not generalized since $>$, $\geq$, $<$, $\leq$ are used to compare numerical values only, while $=$ and $\neq$ are used to compare non-numerical values as well, like strings.

Overall, the reason we applied this SQL keyword generalization pre-processing is to avoid situations where certain common keywords are seen only for a few times or even never seen in the training data set, then the trained model would react poorly to those keywords in the test data set by pulling words from the vocabulary almost randomly.

4 Shuffled Back-Translation

Another idea we utilized to improve the BLEU score is the iterative back-translation as described in Shin (2019) and Zhong et al. (2017).

Back-translation is a simple way of adding synthetic data to the training model by training a target-to-source model, then generating a synthetic source dataset using a monolingual corpus on the target side. The synthetic source dataset and the provided target dataset are augmented to the training datasets to re-train the model. Since no monolingual corpus was provided in our case, we split the original dataset. To address any potential bias, we shuffled the dataset before splitting so that the created monolingual dataset is free from bias.

We also tried a variant of back translation called cyclic translation. The idea simply repeats the step of back-translation. After generating the synthetic source dataset from the provided target dataset, that dataset is used as input to the baseline source-to-target model to generate the synthetic target dataset. The synthetic source dataset and synthetic target dataset are augmented to the training datasets to train the model once again.

The shuffled back-translated model with a high drop-out rate and more number of training steps led to the highest BLEU score on the development dataset as reported in Section 5.

5 Experiment and Results

A lot of diverse models have been trained for our experiments as enumerated below which have been labeled as follows:

1. Baseline (Model 1)

2. Baseline with SQL keyword generalization (Model 2)

3. Baseline with SQL keyword generalization and true-cased input (Model 3)

Model	BLEU score on the dev set
Model 1	7.60
Model 2	9.72
Model 3	10.39
Model 4	11.05
Model 5	10.85
Model 6	10.46
Model 7	11.75
Model 8	9.50
Model 9	12.12

Table 2: Cross validation results with different models.

4. Back-translation with SQL keyword generalization and true-cased input (Model 4)

5. Cyclic-translation with SQL keyword generalization and true-cased input (Model 5)

6. Shuffled back-translation with SQL keyword generalization and true-cased input (Model 6)

7. Back-translation with SQL keyword generalization and true-cased input (higher dropout and more training steps) (Model 7)

8. Cyclic-translation with SQL keyword generalization and true-cased input (higher dropout and more training steps) (Model 8)

9. Shuffled back-translation with SQL keyword generalization and true-cased input and dropout rate = 0.5 (Model 9)

These models have been described in the previous sections. All the notable results are shown in Table 2.

We began our experiment by tuning the hyperparameters of the Seq2seq model in Tensorflow NMT. After repeated experimentation, we selected the parameters for our baseline training model (Model 1) as follows:

1. 4 layered bi-directional encoder
2. Source and target sequence length of 60
3. Adam optimizer
4. 0.001 as the initial learning rate
5. luong10 learning rate decay scheme as described in Tensorflow NMT
6. 12000 training steps
7. 0.4 drop-out rate

The other parameters are set to the default Tensorflow NMT values.

Then, we came up with the idea of SQL keyword generalization and implemented this idea. It turned out to be wonderful and improved the BLEU score significantly (from 7.60 to 9.72). Next, we focused on other possible pre-processing techniques that we could apply. We initially were considering four methods: tokenization, true-casing, linguistic factorization, and byte pair encoding. According to our testing, byte pair encoding, and the combination of these two methods degraded the BLEU score. Linguistic factorization led to high BLEU scores but the removal of the linguistic factors from the generated response again reduced the BLEU score. Tokenization also degrades the performance of the model. After carefully observing the given dataset, we found that it has already been tokenized, so further tokenization is unnecessary. In the end, SQL keyword generalization and true-casing are the two pre-processing techniques that we apply to the model.

Afterwards, we started to think about the steps in the training process that we could improve. We implemented back-translation, and it increased the BLEU score. However, we found this method is likely to introduce an overfitting issue. To be more specific, since we were not given any test data or any dataset analogous to a monolingual corpus, we split the given ground truth file for the development set into two files and used them (one as our "development" ground truth and the other as our "test" ground truth) for the external evaluation during the training. The model achieved a much higher BLEU score on our "development" ground truth than previously recorded but the BLEU score on our "test" ground truth decreased in comparison to that previously recorded.

Then, we came up with three ways to deal with this issue. The first one was the cyclic translation where no extra data (i.e. the monolingual data) is introduced in the training. This new way of training did help with the overfitting issue with a higher BLEU score on our created "test" dataset but failed to improve the BLEU score on the given development set. The second way was to shuffle the

monolingual data used in the back-translation. It solved the overfitting issue but did not achieve a higher BLEU score on the development data either. The last way was to change the values for certain hyper-parameters. For instance, we increased the dropout rate from 0.4 to 0.5 to strengthen regularization. Accordingly, we also increased the number of training steps from 12000 to 20000. We applied the hyper-parameter changes to all three training methods, the original back-translation, cyclic translation, and shuffled back-translation. In the end, the shuffled back-translation model with the new hyper-parameter settings and the two pre-processing practices achieved the highest BLEU score on the development set.

6 Conclusion

Our submitted shuffled back-translation with a drop-out rate of 0.5 and 20000 training steps on Tensorflow NMT gives a BLEU score of 7.47 on the unknown testing dataset and a BLEU score of 12.12 on the development dataset. A further conclusion can be drawn once the Grammar and the Logical Consistency Rate (LCR) scores are released by the organizers. It can be observed that shuffled back-translation with a higher drop-out rate gave a high BLEU score on the development dataset compared to the baseline or the back-translated model with a lower drop-out rate. This suggests that the shuffling of the dataset before back-translation can potentially address the issue of any bias in the datasets. The improved results with increased drop-out suggest that regularization has been effective in this experimental setting. The idea of cyclic translation deserves further exploration. Generalization may be improved on the natural language responses by developing an improved variant of the linguistic factoring approach. The collection of additional training data can also be useful to increase the BLEU score on the unknown test dataset.

References

Saptarashmi Bandyopadhyay. 2019. Factored neural machine translation at loresmt 2019. In *Proceedings of the 2nd Workshop on Technologies for MT of Low Resource Languages*, pages 68–71.

Saptarashmi Bandyopadhyay. 2020. Factored neural machine translation on low resource languages in the covid-19 crisis.

Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. 1997. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery*, 1(1):29–53.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Richard Shin. 2019. Encoding database schemas with relation-aware self-attention for text-to-sql parsers. *arXiv preprint arXiv:1906.11790*.

Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. 2019. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *arXiv preprint arXiv:1909.05378*.

Xiaoshi Zhong, Aixin Sun, and Erik Cambria. 2017. Time expression analysis and recognition using syntactic token types and general heuristic rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 420–429, Vancouver, Canada. Association for Computational Linguistics.