ADVERSARIAL ATTACK FOR SEMANTIC PARSING

A PREPRINT

Jingze Zhang
Department of Computer Science
The Chinese University of Hong Kong
Hong Kong, China
1155107857@link.cuhk.edu.hk

August 16, 2019

ABSTRACT

In recent years, the development of Natural Language Processing (NLP) technology has led to an increasing demand of powerful Semantic Parser. However, in most cases, determining the the efficacy of a semantic parser model can be rather difficult. Especially, there are various kinds of requirements for those models. During the summer research internship, we attempted to attack a text-to-SQL model and find out the influence from the perturbation on input data to the final output. And in this report, we would present the process, result and thinking we get through the research.

1 Introduction

Semantic parsing technology is an important part in the Natural Language Processing technology. And through the developing process of semantic parsing, various kinds of presentation language was chosen to represent the semantic output concisely and accurately. In our research, SQL was chosen for its high conciseness and popularity in database industry. In addition, the Schema-based Graph Neural Network (GNN)[1] model was evaluated and chosen as the model to be attacked. The dataset for training the model and generating adversarial examples is Spider[2] which is relatively large and diverse. We also get the detailed information about Spider and other datasets[2] for comparison in the Table 2.

Table 1: Spider dataset and other datasets[2]

Dataset	Spider	WikiSQL	GeoQuery
# Question	10,181	80,654	877
# SQL	5,693	77,840	247
# Database	200	26,521	247
# Domain	138	-	1
Table/DB	5.1	1	20
ORDER BY	1335	0	20
GROUP BY	1491	0	46
NESTED	844	0	167
HAVING	388	0	9

For the attacking part, we choose the generally used method Fast Gradient Sign Method (FGSM) to generate adversarial examples. And considering the compatibility for the NLP model, we also applied some approximation to ensure the perturbed input is in natural language. However, as the generated question examples may not make sense to people, we also try to manually select some special examples and adjust them until they are grammatical correct.

The final perturbed result was counted and we discovered some potential drawbacks for the model with the statistical data.

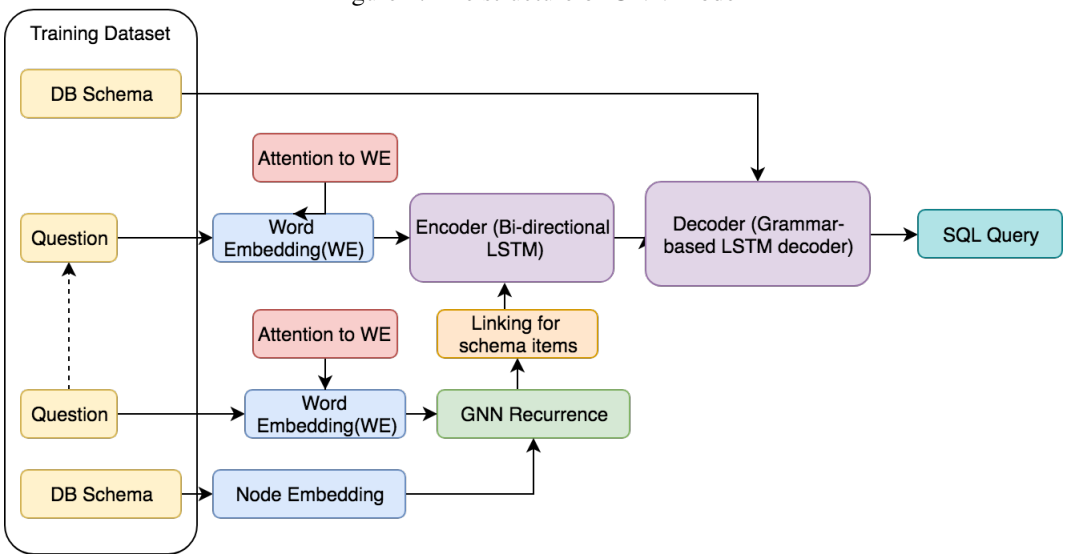
2 Methodology

The main method for attacking process can be divided into three major parts:

- Automatically generate adversarial examples, attack the model, verify the successful examples and count the successful rate
- Manually select the examples with incorrect grammar and re-attack the model with the corrected examples
- Manually select the natural adversarial examples in correct grammar and analyze the difference i the output

Before the attacking process, we firstly evaluate the GNN model with the modified code from the paper[1].

Figure 1: The structure of GNN Model



The model with its structure presented in the Figure 1 has two major features over other models:

- **Graph-based structure**: The GNN model uses graph data format to represent the relation between data items(like a database, a table or a column). In addition to the data items, graph is also used to build the linking table for every pair of database schema, which is crucial in building the weight over each schema.

- **Database schema**: In the training process, database schema is provided for each pair of question-SQL data. Generally, database schema consists of two parts–the schema items which is mainly the name of database, table or column and the reference relation between schema items. In the testing process, the database schema is either extracted from the question provided or dataset used to train the model.

Then we train the model on the Spider dataset for 100 epochs. The final training accuracy of our model and the fully-trained model from the paper[1] are presented in the Table 2.

Table 2: Evaluation result for GNN Model

Model	General Accuracy	Single Accuracy	Multiple Accuracy
Our model	37.9%	49.1%	24.9%
Fully-trained model	40.7%	52.2%	26.8%
SyntaxSQL	18.9%	23.1%	7.0%

Note: single and multiple mean the number of domains involved.

According to the data in the Table 2, in comparison to the previous SyntaxSQL Net[3], the model improves the single accuracy by about only 1 time while improve the multiple accuracy by almost 3 times. The greater increasing in the multiple accuracy indicates that the graph structure optimizes stability for the complex question(according to the multiple accuracy) more apparently.

And in the experiment part(Section 3), we will examine the stability and robustness for the model with adversarial examples.

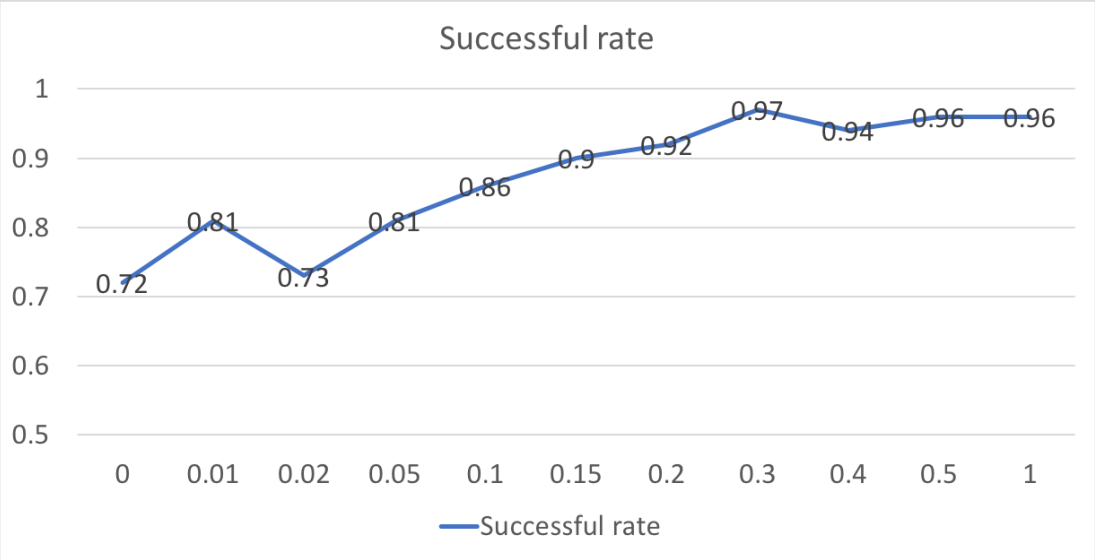
3 Experiments

First, we use the FGSM with approximation mentioned in the methodology part (Section 2). The process is presented as follows:

1. For each input sentence x and expected output y, calculate the gradient of the embedding of x denoted by $grad$.
2. Calculate the perturbed sentence with the formula: $x^{perturb} = x + \epsilon\, sign(grad)$.
3. Find the index i where $grad_i$ reaches its maximum.
4. Find the word closest to $x_i^{perturb}$ denoted by x_i', that is where $|x_i' - x_i^{perturb}|$ reaches its minimum.
5. Create a new sentence according to the original sentence with the ith word replaced by x_i'. The new sentence is denoted by x'.
6. Input x' into the model and get the result y'. Compare y' with y. If $y \neq y'$, the attack is successful, otherwise it is unsuccessful.

For the process above, we have an uncertain coefficient ϵ which determines the perturbation result. Thus, we examined the model with several values for the coefficient ϵ and counted the successful rate for each value. The detailed information about the attacking process is presented with the Figure 2.

Figure 2: The successful rates for values of ϵ



From the Figure 2, we can find that the initial decreasing in the successful rate indicates that in some range of perturbation (for some value of the coefficient ϵ), the model can maintain the stability when there is no perturbation. However, when the perturbation is out of that range, the perturbation will increasingly influence the successful rate until the random factor dominates the influence.

Except the automatic part above, to learn about how the model react with grammatical correct adversarial examples, we also manually select the grammatical correct examples. One representative example is presented as follows.

Example 1

Original sentence:
What ... with **highest** average attendance ? =>
select ... from stadium **order by stadium.average** desc limit 1

Perturbed sentence:
What ... with **greatest** average attendance ? =>
select ... from stadium **group by stadium.stadium_id order by avg (stadium.average)** desc limit 1

We also test some other examples through correcting the generated examples manually and re-atacking the model with these examples. But in this part, in general, the correction of the grammar does not change the attacking result, which indicates that the model will not be easily influenced by the grammar difference. Through this part, we successfully verify the robustness of the model in handling natural language sentences with grammar mistakes.

For the last part of the experiment, we also manually test some cases by replacing the word with its synonyms and antonyms. The result shows that when replacing the word with its synonyms, the model in most cases can always handle this condition and generate the same output. However, when attacking the model through replacing the word with its antonyms, the reaction of the model

is more complicated. The flipping of the sentence semantic may lead to enormous change in the output. One example is as follows.

Example 2

Original sentence:
How much ... **youngest** dog weigh ? =>
select **weight from pets order by pet_age limit 1**

Perturbed sentence:
How much ... **oldest** dog weigh ? =>
select **count (*) from has_pet where has_pet.stuid = ' value '**

For this example, the little difference in the sentence but large difference in the semantic leads to enormous difference in the output. Specially, the syntax structure of the generated SQL query is a large change.

This phenomenon indicates that while the sentence along with the database schema can determine the weight of each item, the change of the keyword which may be weighed too much can lead to the flipping of the sentence meaning. And the large flipping of the sentence meaning makes the influence of the change of one word enlarged. And ultimately, the syntax structure of the output is changed.

4 Conclusion

Through the evaluation process and the attacking experiment, we have verified the robustness of the chosen model. Simultaneously, as mentioned above, we have also discovered some special cases like Example 1 and Example 2.

Generally speaking, the GNN model performs as expected and can handle most of the manually selected examples well. While some special cases indicate that the graph-based data structure along with "schema" have some drawbacks that the weight got with the graph-based structure over the items in the sentence can be hard to examine their validity. In the condition that some word gets weighed too much, little change of that word can lead to enormous change of the output.

References

[1] Ben Bogin, Matt Gardner, and Jonathan Berant. Representing schema structure with graph neural networks for text-to-sql parsing, 2019.

[2] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*, 2018.

[3] Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of EMNLP*. Association for Computational Linguistics, 2018.