LinGAN: an Advanced Model for Code Generating based on Linformer

Hongming Dai

1 Department of Computer Science, Northeastern University, Shenyang, Liaoning, 110000, China
*Corresponding author’s e-mail: 20184542@stu.neu.edu.cn

Abstract—Parsing natural language to corresponding programming language attracts much attention in recent years. Natural Language to SQL (NL2SQL) widely appears in numerous practical Internet applications. Previous solution was to convert the input as a heterogeneous graph which failed to learn good word representation in question utterance. In this paper, we propose a Relation-Aware framework named LinGAN, which has powerful semantic parsing abilities and can jointly encode the question utterance and syntax information of the object language. We also propose the pre-norm residual shrinkage unit to solve the problem of deep degradation of Linformer. Experiments show that LinGAN achieves excellent performance on multiple code generation tasks.

1. Introduction

Natural Language to Programming Language (NL2PL) is the task of converting a natural language utterance to a logical form: machine-understandable representation of its meaning. Recent years, NL2PL has been more and more widely applied to web applications such as Knowledge Based Question Answering (KBMA), Machine Reading Comprehension (MRC) and so on. Currently, NL2PL task poses the great challenge of generalization to unseen additional programming information like database schemas or complex logical semantics in some programming languages[3].

To solve the challenge like schema generalization, model must takes into account both known complex relations and. the good representation of the question utterance. GCNGAN has been proved an efficient semantic parser for the question utterance. GCNGAN has achieved a good semantic extraction performance by transforming the natural language utterance into DAG, and using GCN to learn the graph’s information. However, its ability of learning relation-aware information is questionable. RAT-SQL[3] has achieved the brilliant performance on utilizing complex relations in SQL by encoding the relational structure in the database schema and applying self-attention to combine the reasoning and the natural language. But RAT-SQL’s strategy is too limited in generating SQL and its ability to generate other programming languages needs to be improved.

We propose LinGAN to accomplish the task. Its encoder uses a deep Linformer for backbone while its decoder uses. SmBop for backbone. Encoder aims to extract complex semantic information while decoder aims to capture the target language’s syntax with abstract syntax tree[7]. In order to further improve the effect of parsing semantics and solve deep degradation of the Linformer, we improve the basis of the pre-norm residual unit[4] using pre-norm residual shrinkage unit.

Based on Linformer with pre-norm residual shrinkage unit, the upper limit of the Linformer’s depth has been greatly improved compared to the traditional one. Thus, our model can learn more massive data and have more powerful feature extraction capabilities.
2. Model Architecture

2.1. Generator

In generator, we apply encoder-decoder framework to reduce the interaction between different languages. Encoder-Decoder model provides sufficient flexibility for the whole model because its ends can be many deep learning models. Here, we propose Linformer[2] as both ends. However, traditional transformers are prone to gradient disappearance and gradient explosion with the increasing depth. One solution to this problem is to change the location of the layer normalization[4]. By moving the layer normalization before the residual unit, the model can get better BLEU score on WMT dataset with less cost of model training. However, the single-step property of the pre-norm residual shrinkage unit is easy to forget the distant layers’ information[5]. In order to further improve the effect of parsing semantics and solve deep degradation of the Linformer, we improve the basis of the pre-norm residual unit[4] using pre-norm residual shrinkage unit. Traditional pre-norm residual is as follows:

\[
x_{l+1} = f(y_l) \\
y_l = x_l + \text{ReLU}(x_l; \theta_l) \\
x_{l+1} = x_l + \text{ReLU}(\text{LN}(x_l); \theta_l)
\]

(1)

Here, we introduce soft thresholds to further filter important and unimportant features. Combined with the attention mechanism, the soft threshold will zero the scores of unimportant features, but will increase the scores of important features. Here we propose residual shrinkage unit’s soft thresholds.

\[
\begin{align*}
x - \tau & \quad x > \tau \\
0 & \quad -\tau \leq x \leq \tau \\
x + \tau & \quad x < -\tau
\end{align*}
\]

(2)

To solve the above problem, we propose the model called Linformer[6]. Linformer can turn the computing complexity of the self-attention 0 (n2) to 0 (n). With the linear computational complexity, the self-attention can learn more distant sequences’ dependence information because we can increase the attention span. As is known, each head of the traditional attention is defined as follows:

\[
\text{head}_i=\text{Attention}(QW_i^Q, KW_i^K, VW_i^V)=\text{softmax}\left(\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d_k}}\right)VW_i^V
\]

(3)

Where \(W_i^Q, W_i^K \in \mathbb{R}^{dn \times dm}\). For any \(Q, K, V \in \mathbb{R}^{n \times d}\), \(W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{n \times d}\), for any column vector \(\omega \in \mathbb{R}^n\) of matrix \(VW_i^V\), there exists a low-rank matrix \(P \in \mathbb{R}^{n \times n}\) such that

\[
\Pr(\|\tilde{P}\omega^T - P\omega^T\| < \epsilon \|P\omega^T\|) > 1 - o(1), \text{rank}(\tilde{P}) = \Theta(\log(n))
\]

(4)

Due to the fact of low-rank property of the context mapping matrix \(P\), one straightforward idea is to use singular value decomposition(SVD) to approximate \(P\) with a low-rank matrix \(P'\) based on the Eckart–Young–Mirsky Theorem:

\[
P \approx P_{low} = [u_1, \ldots, u_k] \text{diag}\{\sigma_1, \ldots, \sigma_k\} \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix}
\]

(5)
2.2. Discriminator
In discriminator, we implement SmBop[1] to convert Abstract Syntax Trees to semantics in a bottom-up way, which means convert it from leaf nodes to root nodes. SmBop will first compute the score of all the ASTs on the frontier. After scoring the frontier trees, the representation of trees will be scored and finally the programming language will be generated during AST’s leaves’ initialization. Let $M_{AST}$ be the final encoding vector of the AST that generator generates, and $M_v$ be the semantic vector of the natural language sequence. So the possibility $P$ that the semantic of AST is consistent with the natural language is as follows:

$$ S = M_{AST} W^D M_v + \beta $$

$$ P = \text{softmax} \left( S \right) = \frac{e^{consistent}}{\sum_{i \in S} e^i} $$

According the similarity of two semantics, discriminator will give instant feedback to generator based on Award Function. So that the generator’s generating ability can become better and better.

3. Experiments

3.1 Datasets
- Spider dataset, which includes complex SQL queries, contains 8659 training examples and 1034 evaluation examples.
- Python dataset contains 20000 training examples and 5000 evaluation examples that we manually labeled every single line of programming code with the corresponding natural language.
- Jobs dataset contains 400 training examples and 150 evaluation examples.
- CoNaLa dataset contains 2379 training and 500 evaluation examples.

3.2 Experiment Results

![Graph](image)

Note: All experiments run with 8 NVIDIA Titan V GPUs. All models are optimized by Adam with $\beta_1 = 0.5, \beta_2 = 0.7, s = 10^{-7}$

**TABLE I: Exact Set Match Accuracy of Different Models**

| Datasets | LinGAN | TransGAN | GCNGAN | Seq2Seq |
|----------|--------|----------|--------|---------|
| Spider   | 68.5   | 65.6     | 63.9   | 54.2    |
| Python   | 72.1   | 71.3     | 69.9   | 59.1    |
| Jobs     | 89.9   | 89.4     | 88.2   | 76.8    |
| CoNaLa   | 86.2   | 84.4     | 83.3   | 74.7    |
4. Conclusion And Future Work

In this paper, we propose an advanced Linformer with GAN to accomplish code generating task. The advanced Linformer not only reduces the computational complexity of the traditional transformer’s attention to linear complexity, but also greatly solves the problem of gradient disappearance and gradient explosion through the pre-layer normalization layer, and greatly reduces the sample’s noise through the residual shrinkage mechanism. In the future, we will introduce the Prompt learning paradigm to further enhance the semantic understanding of the model. At the same time, we will further enhance the dataset to improve the discriminator’s ability to discriminator.

References
[1] Ohad Rubin and Jonathan Berant. "SmBoP: Semi-autoregressive Bottom-up Semantic Parsing". In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021.
[2] Ashish Vaswani et al. "Attention is All you Need". In: Annual Conference on Neural Information Processing Systems, NIPS 2017.
[3] Bailin Wang et al. "RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers". In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020.
[4] Qiang Wang et al. "Learning Deep Transformer Models for Machine Translation". In: Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019.
[5] Qiang Wang et al. "Multi-layer Representation Fusion for Neural Machine Translation". In: Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018.
[6] Sinong Wang et al. “Linformer: Self-Attention with Linear Complexity”. In: CoRR(2020)
[7] Pengcheng Yin and Graham Neubig. “A Syntactic Neural Model for General-Purpose Code Generation”. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017.