Rethinking Positional Encoding in Tree Transformer for Code Representation

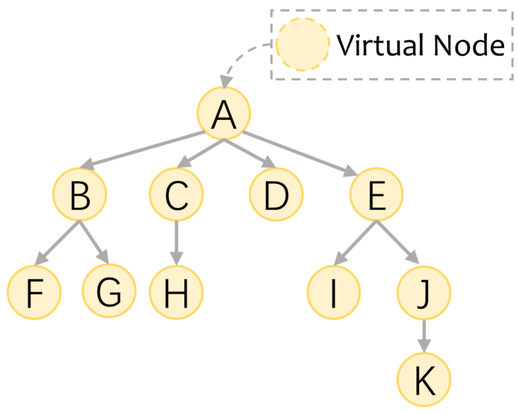
Han Peng, Ge Li*, Yunfei Zhao, Zhi Jin*
Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education;
Institute of Software, EECS, Peking University, Beijing, China

Introduction

Several recent works develop tree Transformers to capture the AST structure in source code, and specifically, novel tree positional encodings have been proposed to incorporate structure inductive bias into Transformer.
In this work, we propose a novel tree Transformer encoding node positions based on our new description method for tree structures. Technically, local and global soft bias is both introduced as positional encodings of our model.

Approach

1) A 2D recursive description for code AST



A : {(1,1)}
B : {(1,1),(1,4)}
.....
J : {(1,1),(4,4),(2,2)}
K : {(1,1),(4,4),(2,2),(1,1)}

Our proposed description for tree structure is defined from the tree root to leaves recursively. Specifically, the position for each node is represented as:

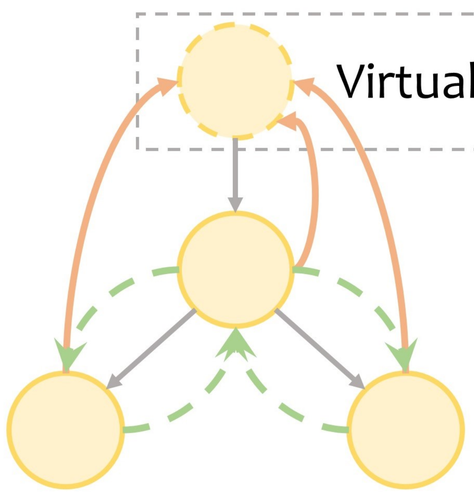
$$\mathscr{F}(x) = \begin{cases} \mathscr{F}(f(x)) + \{(x_i, x_j)\} & if\ x \neq root \\ \{(1,1)\} & if\ x = root \end{cases}$$

Emb

$$H_i = [h(i^1), h(i^2), ..., h(i^n)]$$

The $\mathscr{F}(x)$ is the position description for node x and $f(x)$ is specified as the parent node for it. The x_i is the sibling order of node x and the x_j is the total child number of its parent $f(x)$. The n is the depth of node i in the tree and $h(i^n)$ is the embedding vector of the nth two-dimensional coordinate.

2) Encoding tree positions in local and global



Local-rel:
Global-abs:

$$r_{ij} = \begin{cases} LN(Linear(\sum_h H_i - \sum_h H_j)) & if\ f(i) = j\ \vee\ f(j) = i \\ \vec{0} & if\ f(i) \neq j\ \wedge\ f(j) \neq i \end{cases}$$

$$\gamma_{ij} = (x_i W^Q)(r_{ij} W_r^K)^T + (r_{ji} W_r^Q)(x_j W^K)^T$$

Local-rel

$$a_i = LN(Linear(Concat(H_i)))$$
$$\beta_{ij} = (a_i W_a^Q)(a_j W_a^K)^T$$

Global-abs

$$\alpha_{ij} = (x_i W^Q)(x_j W^K)^T$$

$$A_{ij} = \frac{1}{\sqrt{2}}(\alpha_{ij} + \beta_{ij} + \gamma_{ij}),$$
$$z_i = \sum_{j=1}^{n} \frac{\exp(A_{ij})}{\sum_{j'=1}^{n} \exp(A_{ij'})}(x_j W^V)$$

We feed all nodes into Transformer and integrate positional vectors into self-attention as structure bias by positional encoding.

Experiment

Model	Python					JavaScript				
	MRR (Type)	MRR (Value)	ACC (Type)	ACC (Value)	ACC (All)	MRR (Type)	MRR (Value)	ACC (Type)	ACC (Value)	ACC (All)
Transformer	88.76 ±0.09	54.28 ±0.02	81.91 ±0.13	49.56 ±0.06	59.27 ±0.11	89.51 ±0.01	62.73 ±0.02	82.73 ±0.01	57.40 ±0.02	63.34 ±0.03
(Shaw et al., 2018)	89.05 ±0.04	55.11 ±0.04	82.34 ±0.06	50.48 ±0.08	60.07 ±0.14	90.07 ±0.03	**64.13** ±0.05	83.60 ±0.05	**59.09** ±0.09	64.80 ±0.14
(Shiv and Quirk, 2019)	88.74 ±0.10	54.24 ±0.20	81.81 ±0.14	49.40 ±0.26	59.14 ±0.21	89.78 ±0.10	62.90 ±0.15	83.10 ±0.15	57.48 ±0.20	63.64 ±0.15
(Kim et al., 2020)	91.26 ±0.03	54.79 ±0.02	85.64 ±0.05	50.04 ±0.02	62.15 ±0.02	91.21 ±0.04	63.49 ±0.09	85.12 ±0.06	58.24 ±0.10	65.42 ±0.11
Our model	**91.58** ±0.08	**55.62** ±0.06	**86.15** ±0.12	**51.19** ±0.13	**63.19** ±0.13	**91.63** ±0.03	64.08 ±0.08	**85.91** ±0.06	58.99 ±0.13	**66.48** ±0.14
w/o first dim.	88.81 ±0.11	54.63 ±0.09	81.90 ±0.17	49.86 ±0.14	59.36 ±0.26	89.91 ±0.06	63.25 ±0.22	83.28 ±0.10	57.97 ±0.28	64.02 ±0.29
w/o second dim.	90.78 ±0.02	55.10 ±0.08	84.87 ±0.02	50.52 ±0.10	62.04 ±0.07	91.08 ±0.06	63.50 ±0.10	85.02 ±0.10	58.30 ±0.13	65.47 ±0.17
w/o Global-Abs.	91.40 ±0.09	55.25 ±0.15	85.88 ±0.13	50.68 ±0.21	62.70 ±0.23	91.57 ±0.03	63.95 ±0.09	85.84 ±0.04	58.84 ±0.13	66.34 ±0.11
w/o Local-Rel.	91.35 ±0.07	55.36 ±0.03	85.81 ±0.12	50.81 ±0.08	62.75 ±0.15	91.18 ±0.07	63.51 ±0.15	85.25 ±0.11	58.28 ±0.18	65.67 ±0.12

Table 1: All results on code completion

Model	Python			JavaScript		
	Bleu	F1	Acc	Bleu	F1	Acc
Transformer	56.29±0.06	33.95±0.28	18.29±0.44	56.51±0.13	22.62±0.59	11.66±0.34
(Shaw et al., 2018)	56.86±0.30	35.58±0.48	19.51±0.41	56.75±0.13	23.22±0.73	11.32±1.78
(Shiv and Quirk, 2019)	56.50±0.19	34.54±0.38	18.89±0.48	57.04±0.23	24.01±0.21	11.53±0.53
(Kim et al., 2020)	57.02±0.17	35.55±0.41	19.29±0.35	57.33±0.38	23.47±0.28	12.07±0.74
Our model	**57.28**±0.16	**36.00**±0.10	**19.97**±0.17	**57.72**±0.19	**25.03**±0.37	**13.21**±0.24
w/o first dim.	57.10±0.14	35.94±0.21	19.54±0.20	57.40±0.25	24.22±0.65	12.92±0.27
w/o second dim.	56.89±0.15	35.36±0.10	19.24±0.48	57.27±0.09	24.30±0.29	12.66±0.69
w/o Global-Abs.	56.83±0.14	35.80±0.29	19.47±0.10	57.27±0.32	23.86±0.10	12.30±0.65
w/o Local-Rel.	56.95±0.10	35.36±0.37	19.29±0.66	57.50±0.05	23.74±0.18	13.13±0.09

Table 2: All results on code summarization

We focus on two tasks of code representation learning: code completion and summarization.
Our model substantially outperforms baselines almost in all metrics of both tasks. Besides, we find that both two dimensions in the coordinate list are helpful and also confirm the benefits of fusing the local and global paradigms.