Integrating Tree Path in Transformer for Code Representation

Han Peng[1], Ge Li[1], Wenhan Wang[1], Yunfei Zhao[1], Zhi Jin[1]

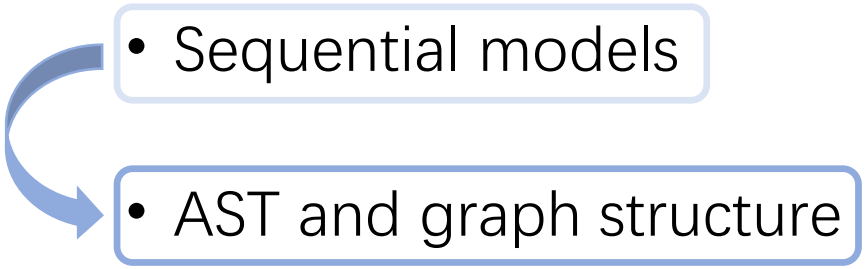
[1]Peking University

Machine Learning for Code

- Aims at building models to learn semantic embedding of code
  - Sequential models

Machine Learning for Code

- Aims at building models to learn semantic embedding of code



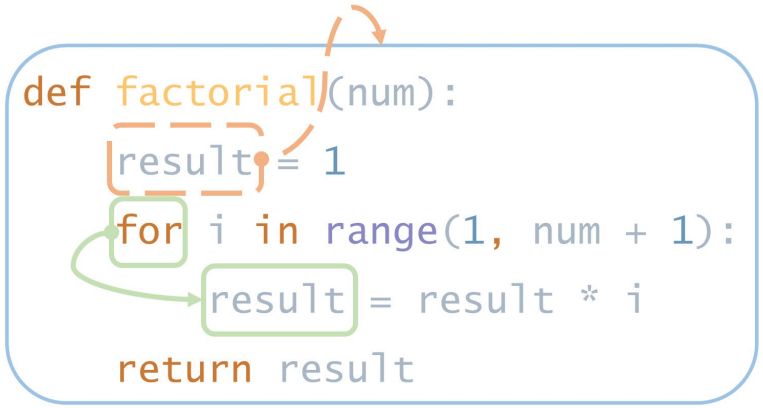
Machine Learning for Code

- Aims at building models to learn semantic embedding of code
  - Sequential models
  - AST and graph structure
  - Graph Neural Network

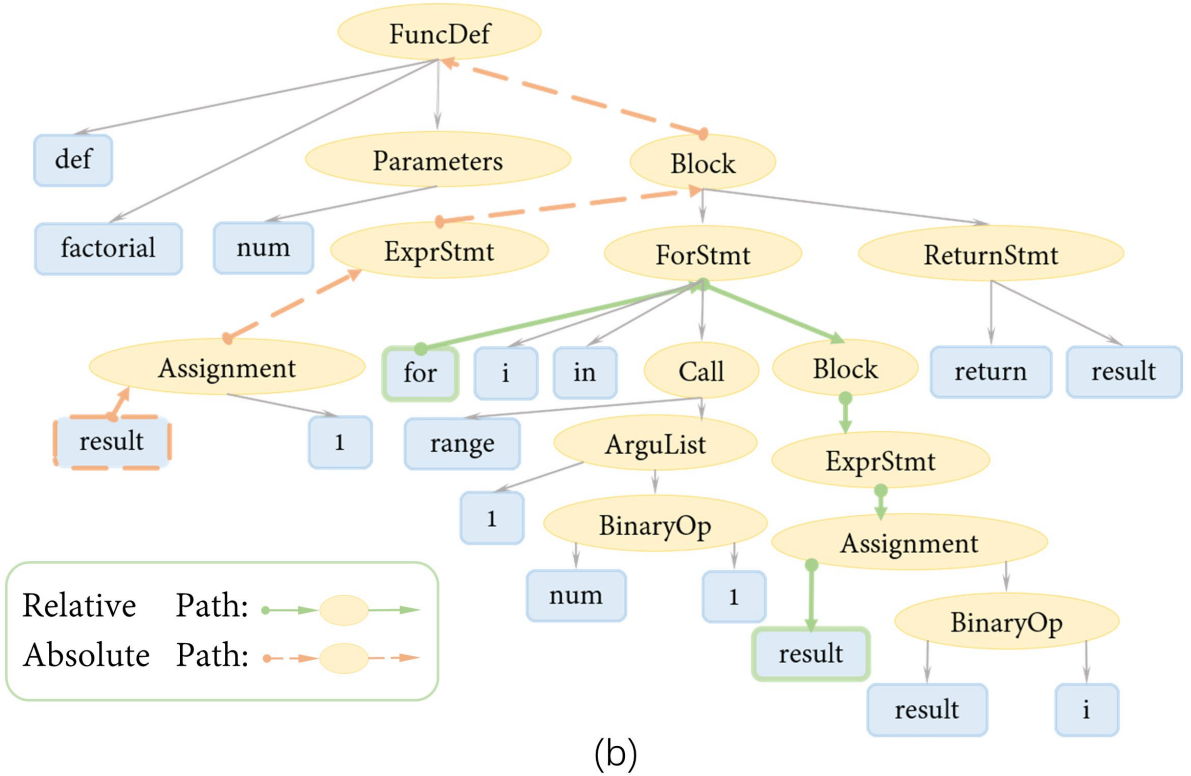
Machine Learning for Code

- Aims at building models to learn semantic embedding of code
  - Sequential models
  - AST and graph structure
  - Graph Neural Network
  - Structural bias Transformer

Two kinds of Tree Path in AST



(a)



(b)

We integrate two kinds of paths into the unified Transformer framework, drawing inspiration from positional encodings.

北京大学 PEKING UNIVERSITY

Vanilla Transformer

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}})V$$

$$z_i = \sum_{j=1}^{n} \frac{\exp(\alpha_{ij})}{\sum_{j'=1}^{n} \exp(\alpha_{ij'})}(x_j W^V),\ where\ \alpha_{ij} = \frac{1}{\sqrt{d}}(x_i W^Q)(x_j W^K)^T$$

Vanilla Transformer

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}})V$$

$$z_i = \sum_{j=1}^{n} \frac{\exp(\alpha_{ij})}{\sum_{j'=1}^{n} \exp(\alpha_{ij'})}(x_j W^V),\ where\ \alpha_{ij} = \frac{1}{\sqrt{d}}(x_i W^Q)(x_j W^K)^T$$

relative positional encoding[1]

$$z_i = \sum_{j=1}^{n} \frac{\exp(\alpha_{ij})}{\sum_{j'=1}^{n} \exp(\alpha_{ij'})}(x_j W^V + a_{ij}^V),\ where\ \alpha_{ij} = \frac{1}{\sqrt{d}}(x_i W^Q)(x_j W^K + a_{ij}^K)^T$$

1)Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani.
Self-attention with relative position representations

Vanilla Transformer

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}})V$$

$$z_i = \sum_{j=1}^{n} \frac{\exp(\alpha_{ij})}{\sum_{j'=1}^{n} \exp(\alpha_{ij'})}(x_j W^V),\ where\ \alpha_{ij} = \frac{1}{\sqrt{d}}(x_i W^Q)(x_j W^K)^T$$

relative positional encoding

$$z_i = \sum_{j=1}^{n} \frac{\exp(\alpha_{ij})}{\sum_{j'=1}^{n} \exp(\alpha_{ij'})}(x_j W^V + a_{ij}^V),\ where\ \alpha_{ij} = \frac{1}{\sqrt{d}}(x_i W^Q)(x_j W^K + a_{ij}^K)^T$$

absolute positional encoding[2]

$$\alpha_{ij} = \frac{1}{\sqrt{2d}}(x_i W^Q)(x_j W^K)^T + \frac{1}{\sqrt{2d}}(p_i U^Q)(p_j U^K)^T$$

2) Guolin Ke, Di He, and Tie-Yan Liu.
Rethinking the positional encoding in language pre-training

Approaches

- Relative Path Encoding
  - 1) $r_{ij} = GRU_r(Path_{x_i \to x_j})$
  - 2) $\alpha_{ij} = \frac{1}{\sqrt{d}}[(x_i W^Q)(x_j W^K + \underbrace{r_{ij} W_r^K}_{1})^T]$
  - 3) $z_i = \sum_{j=1}^{n} \frac{\exp \alpha_{ij}}{\sum_{j'=1}^{n} \exp \alpha_{ij}} (x_j W^V + \underbrace{r_{ij} W_r^V}_{2})$
- Absolute Path Encoding
  - 4) $a_i = GRU_a(Path_{x_i \to root})$
  - 5) $\alpha_{ij} = \frac{1}{\sqrt{d}}[(x_i W^Q)(x_j W^K)^T + (a_i W_a^Q)(a_j W_a^K)^T]$
- Combine them further
  - 6) $\alpha_{ij} = \frac{1}{\sqrt{d}}[(x_i W^Q)(x_j W^K + \underbrace{r_{ij} W_r^K}_{1})^T + (a_i W_a^K)(a_j W_a^V)^T]$

北京大学 PEKING UNIVERSITY

Approaches

- Relative Path Encoding
  - 1) $r_{ij} = GRU_r(Path_{x_i \to x_j})$
  - 2) $\alpha_{ij} = \frac{1}{\sqrt{d}}[(x_i W^Q)(x_j W^K + \underbrace{r_{ij} W_r^K}_{1})^T]$
  - 3) $z_i = \sum_{j=1}^{n} \frac{\exp \alpha_{ij}}{\sum_{j'=1}^{n} \exp \alpha_{ij}} (x_j W^V + \underbrace{r_{ij} W_r^V}_{2})$
- Absolute Path Encoding
  - 4) $a_i = GRU_a(Path_{x_i \to root})$
  - 5) $\alpha_{ij} = \frac{1}{\sqrt{d}}[(x_i W^Q)(x_j W^K)^T + (a_i W_a^Q)(a_j W_a^K)^T]$
- Combine them further
  - 6) $\alpha_{ij} = \frac{1}{\sqrt{d}}[(x_i W^Q)(x_j W^K + \underbrace{r_{ij} W_r^K}_{1})^T + (a_i W_a^K)(a_j W_a^V)^T]$

TPTrans

(Only Rel:[2],[3])

Approaches

- Relative Path Encoding
  - 1) $r_{ij} = GRU_r(Path_{x_i \to x_j})$
  - 2) $\alpha_{ij} = \frac{1}{\sqrt{d}}[(x_i W^Q)(x_j W^K + \underbrace{r_{ij} W_r^K}_{1})^T]$
  - 3) $z_i = \sum_{j=1}^{n} \frac{\exp \alpha_{ij}}{\sum_{j'=1}^{n} \exp \alpha_{ij}} (x_j W^V + \underbrace{r_{ij} W_r^V}_{2})$
- Absolute Path Encoding
  - 4) $a_i = GRU_a(Path_{x_i \to root})$
  - 5) $\alpha_{ij} = \frac{1}{\sqrt{d}}[(x_i W^Q)(x_j W^K)^T + (a_i W_a^Q)(a_j W_a^K)^T]$
- Combine them further
  - 6) $\alpha_{ij} = \frac{1}{\sqrt{d}}[(x_i W^Q)(x_j W^K + \underbrace{r_{ij} W_r^K}_{1})^T + (a_i W_a^K)(a_j W_a^V)^T]$

TPTrans
(Only Rel:[2],[3])

TPTrans-α
(Rel+Abs:[6],[3])

北京大学
PEKING UNIVERSITY

Result: Code Summarization

Model	Python			Ruby			Javascript			Go		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Code2seq	35.79	24.85	29.34	23.23	10.31	14.28	30.18	19.88	23.97	52.30	43.43	47.45
Great	35.09	31.62	33.26	24.66	22.25	23.39	31.25	26.87	28.89	50.02	46.52	48.21
XLNet	37.39	32.01	34.49	29.88	25.89	27.74	33.33	26.86	29.75	51.79	47.58	49.60
Code Transformer	36.41	33.68	34.99	31.46	24.50	27.55	35.07	29.65	32.13	55.09	48.05	51.33
TPTrans	38.45	33.63	35.88	**32.70**	**27.75**	**30.02**	33.47	28.27	30.65	**56.19**	**51.14**	**53.54**
TPTrans-α	**38.48**	**33.99**	**36.09**	32.54	26.77	29.38	**34.06**	**28.42**	**30.99**	56.00	50.97	53.37
TPTrans※	**38.76**	34.66	**36.59**	32.40	27.63	29.82	**33.83**	28.37	30.86	**55.79**	51.05	53.30
TPTrans-α※	38.39	**34.70**	36.45	**33.07**	**28.34**	**30.52**	33.68	**28.95**	**31.14**	55.67	**51.31**	**53.39**

※ means a bigger parameter setting

北京大学 PEKING UNIVERSITY

Analyze: Relationship of two paths

Model	Python			Ruby			Javascript			Go		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
TPTrans	**38.76**	**34.66**	**36.59**	**32.40**	**27.63**	**29.82**	**33.83**	**28.37**	**30.86**	**55.79**	**51.05**	**53.30**
w/o Rel. in *V*	36.74	34.06	35.35	29.96	25.87	27.76	31.93	27.31	29.43	54.85	49.48	52.03
w/o Rel. in *K,V*	33.92	29.50	31.55	24.78	20.32	22.33	27.94	23.08	25.28	49.19	45.87	47.47
TPTrans-α	**38.39**	**34.70**	**36.45**	**33.07**	**28.34**	**30.52**	**33.68**	**28.95**	**31.14**	**55.67**	**51.31**	**53.39**
w/o Rel. in *V*	37.68	33.86	35.67	30.44	25.27	27.58	32.93	28.29	30.43	54.00	50.81	52.36
w/o Rel. in *K,V*	36.67	31.43	33.85	30.05	22.68	25.83	31.17	26.11	28.41	52.07	49.62	50.82

Analyze: Relationship of two paths

Model	Python			Ruby			Javascript			Go		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
TPTrans	**38.76**	**34.66**	**36.59**	**32.40**	**27.63**	**29.82**	**33.83**	**28.37**	**30.86**	**55.79**	**51.05**	**53.30**
w/o Rel. in *V*	36.74	34.06	35.35	29.96	25.87	27.76	31.93	27.31	29.43	54.85	49.48	52.03
w/o Rel. in *K,V*	33.92	29.50	31.55	24.78	20.32	22.33	27.94	23.08	25.28	49.19	45.87	47.47
TPTrans-α	**38.39**	**34.70**	**36.45**	**33.07**	**28.34**	**30.52**	**33.68**	**28.95**	**31.14**	**55.67**	**51.31**	**53.39**
w/o Rel. in *V*	37.68	33.86	35.67	30.44	25.27	27.58	32.93	28.29	30.43	54.00	50.81	52.36
w/o Rel. in *K,V*	36.67	31.43	33.85	30.05	22.68	25.83	31.17	26.11	28.41	52.07	49.62	50.82

Summary

- We propose TPTrans, which integrates path encoding in Transformer for representation learning code.
- We integrate two kinds of paths into the unified Transformer framework and investigate the interaction between them.
- We confirm much feature overlap exists between the two paths.

Code is publicly available at
https://github.com/AwdHanPeng/TPTrans

Thank you!