since 1995

Data Science Natural Language Processing NLP Papers Summary

Day 148: NLP Papers Summary – A Transformer-Based Approach For Source Code Summarization

By Ryan 27th May 2020 No Comments

Objective and Contribution

Utilised a simple transformer-based model with relative position representations and copy attention mechanism to generate SOTA results for source code summarisation. We found that the absolute encoding of source code tokens' position hinders the performance of summarisation whereas the relative encoding significantly improves the performance.

WHAT IS SOURCE CODE SUMMARISATION?

The objective is to encode source code and generate a readable summary that describ he functionality of the program.

Datasets

We have two evaluation datasets: the Java and Python dataset from GitHub as shown below. Our evaluation metrics are BLEU, METEOR, and ROUGE-L.

Dataset	Java	Python
Train	69,708	55,538
Validation	8,714	18,505
Test	8,714	18,502
Unique tokens in code	66,650	307,596
Unique tokens in summary	46,895	56,189
Avg. tokens in code	120.16	47.98
Avg. tokens in summary	17.73	9.48

Table 1: Statistics of the experiment datasets. We thank the authors of Wei et al. (2019) for kindly sharing the Python dataset splits. The Java dataset splits are publicly available.

Methodology

Our proposed model is the vanilla Transformer. We encoded the code and summary as sequence of embeddings. The vanilla Transformer has stacked of multi-head attention and linear transformation layers in the encoder and decoder. We also included the copy attention in the Transformer to allow the model the ability to copy rare tokens from the source code.

POSITION REPRESENTATIONS

Here, we explored the absolute position encoding on the sequential order of source code tokens and the pairwise relationship encoding in Transformer. The absolute position encoding aims to capture the order information of source tokens, however, we show that the order information is actually not helpful in learning source code representations and leads to bad summarisation. We found that it is the mutual interactions between tokens that influence the meaning of the source code, which it's why we explore pairwise relationship encoding. To capture this pairwise relationships between input tokens, we capture the relative positional representations of two position i and j for each token.

Results

As shown below, our full model outperformed all the baseline models. In fact, the base model trained on the dataset without the CamelCase and snake_case code token processing, outperformed all baseline models except on the ROUGE-L metric. Our baseline models didn't incorporate copy attention mechanism and we shown that the copy attention mechanism does improve the performance of our full model.

Methods	Java			Python		
	BLEU	METEOR	ROUGE-L	BLEU	METEOR	ROUGE-L
CODE-NN (Iyer et al., 2016)	27.60	12.61	41.10	17.36	09.29	37.81
Tree2Seq (Eriguchi et al., 2016)	37.88	22.55	51.50	20.07	08.96	35.64
RL+Hybrid2Seq (Wan et al., 2018)	38.22	22.75	51.91	19.28	09.75	39.34
DeepCom (Hu et al., 2018a)	39.75	23.06	52.67	20.78	09.98	37.35
API+CODE (Hu et al., 2018b)	41.31	23.73	52.25	15.36	08.57	33.65
Dual Model (Wei et al., 2019)	42.39	25.77	53.61	21.80	11.14	39.45
Our models and ablation study						
Base Model	43.41	25.91	52.71	31.08	18.57	44.31
Full Model	**44.58**	**26.43**	**54.76**	**32.52**	**19.77**	**46.73**
Full Model w/o Relative Position	44.26	26.23	53.58	31.38	18.69	44.68
Full Model w/o Copy Attention	44.14	26.34	53.95	31.64	19.17	45.42

Table 2: Comparison of our proposed approach with the baseline methods. The results of the baseline methods are directly reported from (Wei et al., 2019). The "Base Model" refers to the vanilla Transformer (uses absolute position representations) and the "Full Model" uses relative position representations and includes copy attention.

Ablation studies

IMPACT OF POSITION REPRESENTATION

Table 3 below showcase the performance of performing absolute position encoding on source and targets. It showcase the decrease in performance when include the absolute position encoding. Table 4 showcase the benefit of learning pairwise relationship between source code tokens. We experimented with different clipping distance and whether we should include the bidirectional information. The performance of different clipping distance are very similar to the performance of our full model and models that include the directional information outperformed the ones that didn't.

Source	Target	BLEU	METEOR	ROUGE-L
✓	✓	43.41	25.91	52.71
✓	✗	42.34	24.74	50.96
✗	✓	**43.59**	**26.00**	**52.88**
✗	✗	41.85	24.32	50.87

Table 3: Ablation study on absolute positional representations using the "Base Model" on the Java dataset.

k	Directional	BLEU	METEOR	ROUGE-L
8	✓	44.22	26.35	53.86
	✗	42.61	24.67	51.10
16	✓	44.14	26.34	53.95
	✗	44.06	26.31	53.51
32	✓	**44.55**	**26.66**	**54.30**
	✗	43.95	26.28	53.24
2^i	✓	44.37	26.58	53.96
	✗	43.58	25.95	52.73

Table 4: Ablation study on relative positional representations (in encoding) for Transformer. While 8, 16, and 32 represents a fixed relative distance for all the layers, 2^i (where $i = 1, \ldots, L$; $L = 6$) represents a layer-wise relative distance for Transformer.

VARYING MODEL SIZE AND NUMBER OF LAYERS

Our results below showcase that a deeper model (more layers) performs better than a wider model (more neurons per layer). We suspect that deeper model is more beneficial in source code summarisation as it depends more on semantic information than syntactic.

QUALITATIVE ANALYSIS

Our qualitative example below showcase that the copy attention mechanism enabled model to generate shorter summaries with more appropriate keywords. We observed that frequent tokens in the source code has a higher copy probability when we use relative position representations.

Conclusion and Future Work

A potential future work could be to incorporate the code structure into Transformer and apply it to other code sequence generation tasks such as the generation of commit messages for source code changes.

Source: **https://arxiv.org/pdf/2005.00653.pdf**

Ryan

Data Scientist

Previous Post

Next Post

Day 147: NLP Papers Summary - Two Birds, One Stone: A Simple, Unified Model for Text Generation from Structured and Unstructured Data

Day 149: NLP Papers Summary - MOOCCube: A Large-scale Data Repository for NLP Applications in MOOCs

Data Science Natural Language Processing NLP Papers Summary

Day 365: NLP Papers Summary – A Survey on Knowledge Graph Embedding

Ryan
30th December 2020

Data Science **Implementation** **Natural Language Processing**

Day 364: Ryan's PhD Journey – OpenKE-PyTorch Library Analysis + code snippets for 11 KE models

Ryan
29th December 2020

Data Science Ryan's PhD Journey

Day 363: Ryan's PhD Journey – Literature Review – Knowledge Acquisition – 1st Passes XIV

Ryan
28th December 2020