北京大学 PEKING UNIVERSITY

高可信软件技术教育部重点实验室
Key Laboratory of High Confidence Software Technologies (PKU), Ministry of Education

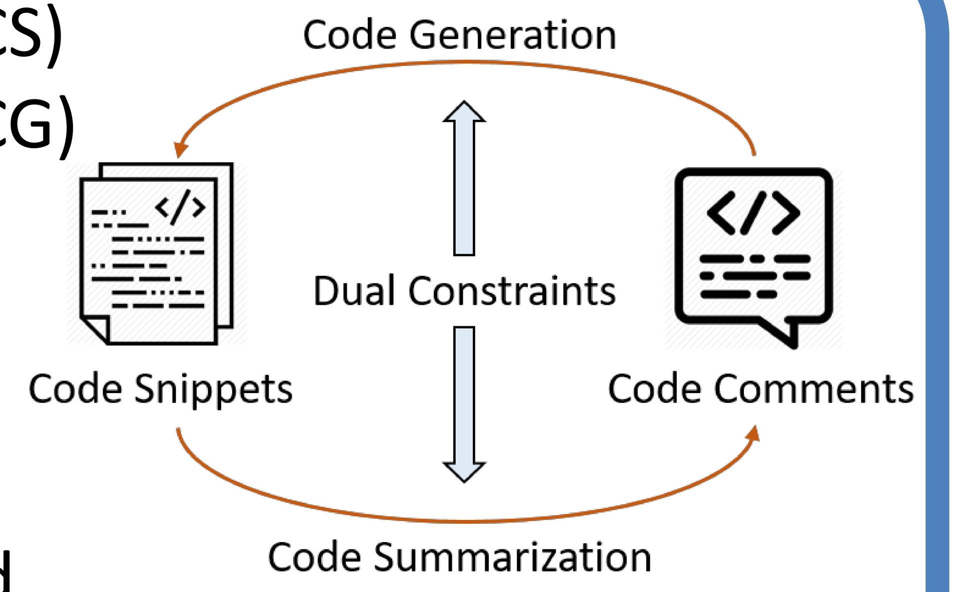
Code Generation as a Dual Task of Code Summarization

Bolin Wei[1], Ge Li[1], Xin Xia[2], Zhiyi Fu[1], Zhi Jin[1]

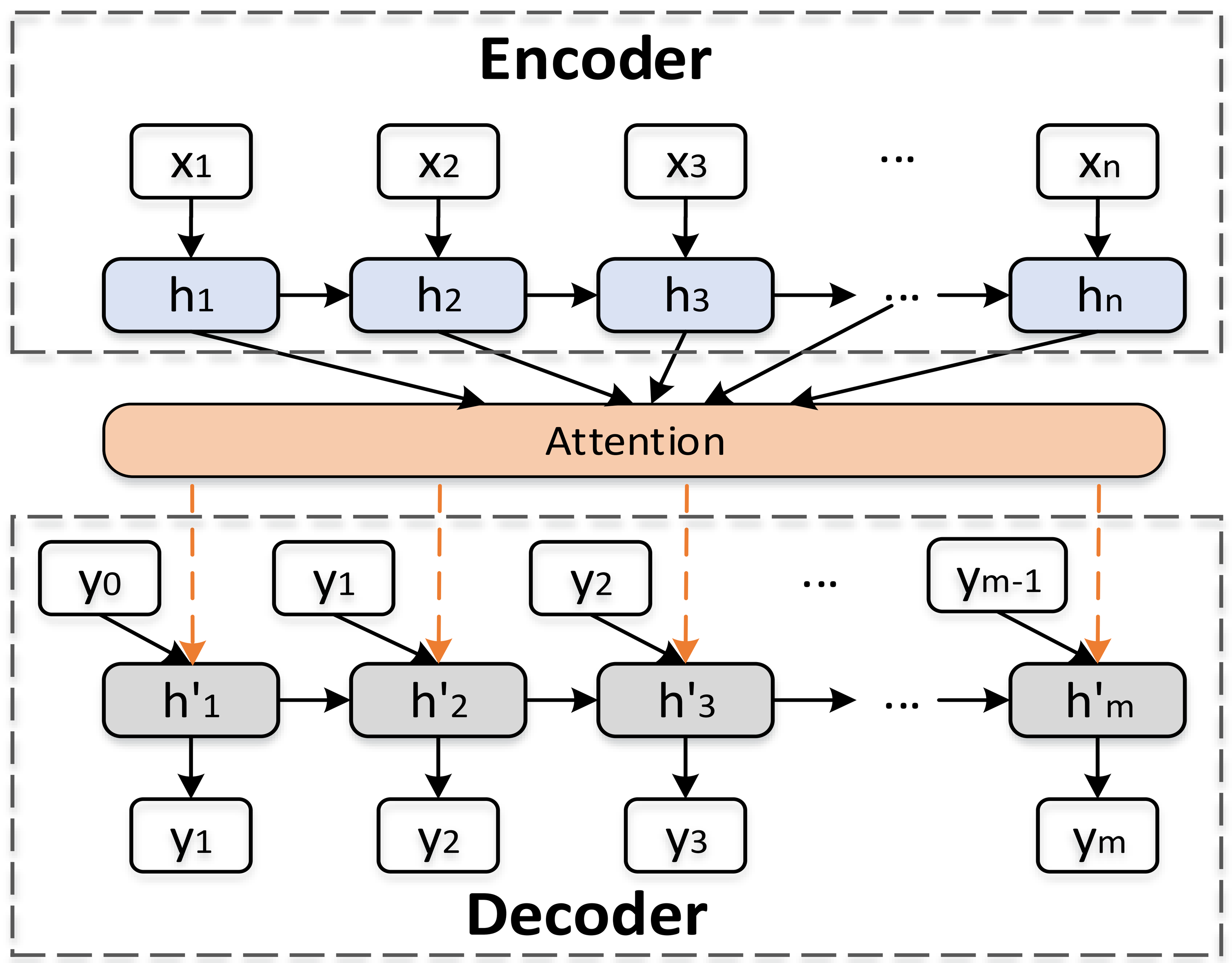
[1]Key Laboratory of High Confidence Software Technologies (Peking University) Ministry of Education, China;
[1]Software Institute, Peking University, China [2]Faculty of Information Technology, Monash University, Australia

Introduction

Code summarization (CS) and code generation (CG) are two crucial tasks in the field of software development. None of the previous studies before have considered the relations between the two tasks or exploited the relations to improve each other. Thus, we design a dual learning framework to train a CS and a CG model simultaneously to exploit the duality of them. Besides a probabilistic correlation, we design a novel dual constraint about attention mechanism.



CS & CG Model



Dual Constraints

Probability:

$$p(x|y) * p(y) = p(y|x) * p(x)$$

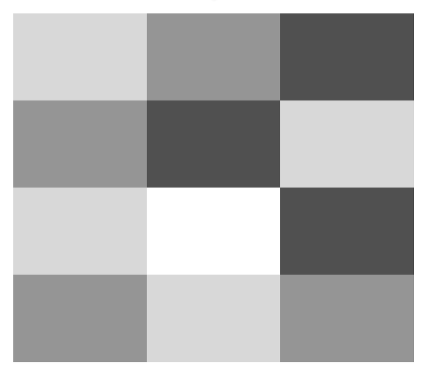
Lagrange multipliers

$$\log p(x) + \log p(y|x) - \log p(x|y) - \log p(y)$$

Attention Weights:

$$b_i = softmax(A_{xy}^i) \text{ from CS}$$
$$b_i' = softmax(A_{yx}^i) \text{ from CG}$$

Using JS divergence to constrain the distance



Algorithm

Algorithm 1 Algorithm Description

Input: Language models $\hat{P}(x)$ and $\hat{P}(y)$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$;
hyper parameters λ_{dual1}, λ_{dual2}, λ_{att1} and λ_{att2};
optimizers *opt*1 and *opt*2
repeat
Get a minibatch of k pairs $\langle(x_i, y_i)\rangle_{i=1}^{k}$;
Calculate the gradients for θ_{xy} and θ_{yx}.

$$G_{xy} = \nabla_{\theta_{xy}}(1/k)\sum_{i=1}^{k}[l_{xy} + \lambda_{dual1} l_{dual} + \lambda_{att1} l_{att}];$$

$$G_{yx} = \nabla_{\theta_{yx}}(1/k)\sum_{i=1}^{k}[l_{yx} + \lambda_{dual2} l_{dual} + \lambda_{att2} l_{att}];$$

Update θ_{xy} and θ_{yx}
$\theta_{xy} \leftarrow opt_1(\theta_{xy}, G_{xy})$, $\theta_{yx} \leftarrow opt_2(\theta_{yx}, G_{yx})$
until models converged

CS Results

Table 2: The overall performance of our CS models compared with baselines

Methods	Java BLEU	Java METEOR	Java ROUGE-L	Python BLEU	Python METEOR	Python ROUGE-L
CODE-NN	27.60	12.61	41.10	17.36	9.288	37.81
DeepCom	39.75	23.06	52.67	20.78	9.979	37.35
Tree2Seq	37.88	22.55	51.50	20.07	8.957	35.64
RL+Hybrid2Seq	38.22	22.75	51.91	19.28	9.752	39.34
API+CODE	41.31	23.73	52.25	15.36	8.571	33.65
Basic Model	41.01	23.26	51.64	20.47	10.38	38.77
Dual Model	**42.39**	**25.77**	**53.61**	**21.80**	**11.14**	**39.45**

Our dual model obviously outperforms all the baselines on three metrics at the same time. The results of Wilcoxon Rank Sum test and Cliff's delta both indicate the improvements are significant.

CG Results

Table 3: BLEU scores and percentage of valid code (PoV) on CG task

Methods	Java BLEU	Java PoV	Python BLEU	Python PoV
SEQ2TREE	13.80	22.6%	4.472	22.7%
Basic Model	10.86	19.6%	10.43	41.8%
Dual Model	**17.17**	**27.4%**	**12.09**	**51.9%**

Although the BLEU score is very low, dual training can still improve the performance of individually trained basic model, proving the effectiveness of dual training.

Conclusion

We aim to build a framework which uses the CG as a dual task for the CS. We applied two constraints on the dual training loss function. Experimental results show that the dual training process helps the CS and CG model surpass the existing SOTA methods on two datasets.