Table 1: Percentage of valid code on CG task.

Methods	Java	Python
SEQ2TREE	22.6%	13.9%
Basic	19.6%	17.5%
Dual	**27.4%**	**25.2%**

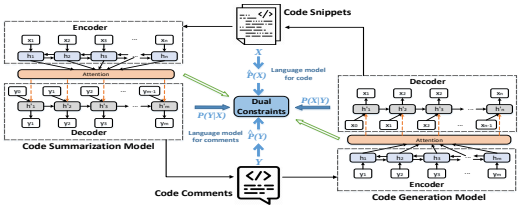

Figure 1: The overall dual training framework.

We thank all reviewers for the comments and feedback.

R1-Key Novelty. The joint training framework is proposed for the first time in the field of program modeling in the paper. Besides, we design a new dual regularization term on the attention mechanism that has never been proposed in other machine learning fields. The probabilistic constraint of Eqn. 8 is similar to the previous work [Xia et al. 2017].

R1-Grammar Constraints. Due to limited time, we evaluated SNM [Yin and Neubig, 2017] on Python dataset. SNM explicitly introduces the constraints of grammar rules when generating ASTs. The BLEU score for SNM is 10.62 and similar to our Basic model, indicating that the CG task on this dataset is very challenging. In particular, all prediction of SNM is valid, whereas the percentage of valid code generated by the dual model is low (Table 1). Hence, it is advantageous for the current dual model to constrain the generated code to satisfy the grammar rules, which will increase the percentage of valid code. Noting that the dual learning is a paradigm for joint training CS and CG. Integrating grammar rules into one model does not affect the dual relationship between the two models, and we leave it as our future work. In the paper, we only focus on the use of duality between CS and CG. We will add the comparison and discuss the future work on our dual model in the paper.

R2-Evaluation Metric. To evaluate how much of the generated code is valid, we calculate the percentage of code that can be parsed into an AST, as shown in Table 1. Dual training can increase the percentage of valid code. 27.4% of the predicted code of the dual model in Java is valid, and 25.2% of the prediction in Python is valid. We will add the details in our paper.

R2-Grammar Constraints. Please read our reply to Reviewer #1 (Grammar Constraints).

R2-Python Dataset. The original data has about 110K parallel pairs (after duplicate example removal), and we mistyped the number in our paper, which can be confirmed from Barone and Sennrich [2017]. Wan et al. [2018] used the dataset in their experiments and published their code on GitHub. We processed the data according to their code. Specifically, we limited the maximum length of code and comments to 100 and 50 respectively, and filtered out samples that cannot be parsed into ASTs. Finally, we used the 55k of data to train CS and CG models (Table 1 in the paper). This training data is not enough for training a language model for comments. We will highlight the details in the paper.

R2-Dual Training Framework. Our dual framework is shown in Figure 1. Language models are only used to calculate marginal probabilities $\hat{P}(x)$ and $\hat{P}(y)$. Except for Tree2Seq and RL+Hybrid2Seq (They used Word2vec to pretrain their token embeddings), the token embeddings for other models are randomly initialized. Warm-starting means that we pretrained CS and CG models separately, then applied dual constraints to the two models for joint training. Pretraining is common in previous work [Xia et al. 2017; Li et al. 2018], and we found it can speed up the convergence process of joint training. Without pretraining, we can also get results beyond baselines. We will add the details in the paper.

R4-Key Novelty. Please read our reply to Reviewer #1 (Key Novelty).

R4-Basic Model vs. Baselines. We use a bidirectional LSTM (bi-LSTM) as the encoder and use beam search in the inference process, which is different from the Attention-based Seq2Seq in [Hu et al. 2018a]. Besides, the attention calculation in our model (Eqn. 2) is different from the method in [Hu et al. 2018a]. They did not report hyperparameters of baselines in their paper; hence, for our basic model, we use the same hyperparameters as our dual model.

R4-Parameters in All Models. Since CS and CG models are trained at the same time and the parameters of the two models are separate after the joint training, i.e., the two models solve their respective tasks separately after the joint training, the number of parameters of each dual model is the same as that of the basic model. The number of parameters for all models on Java CS task is as follows: CODE-NN 34M, DeepCom 53M (We use a bi-LSTM as the encoder to ensure the number of parameters is comparable to that of our model.), Tree2Seq 52M, RL+Hybrid2Seq 80M, API+CODE 80M (We set the embeddings and GRU states to 512 dimensions.), Basic model 53M and Dual model 53M. The relationship between the number of all models' parameters is consistent in Java and Python experiments.

R4-Python Dataset. Please read our reply to Reviewer #2 (Python Dataset).

R4-Training Details. We select the best model according to the performance of the validation set after joint training thirty epochs in both Java and Python experiments for our dual model. We did not use the test data to tune parameters.