SampleFix: Learning to Correct Programs by Efficient Sampling of Diverse Fixes

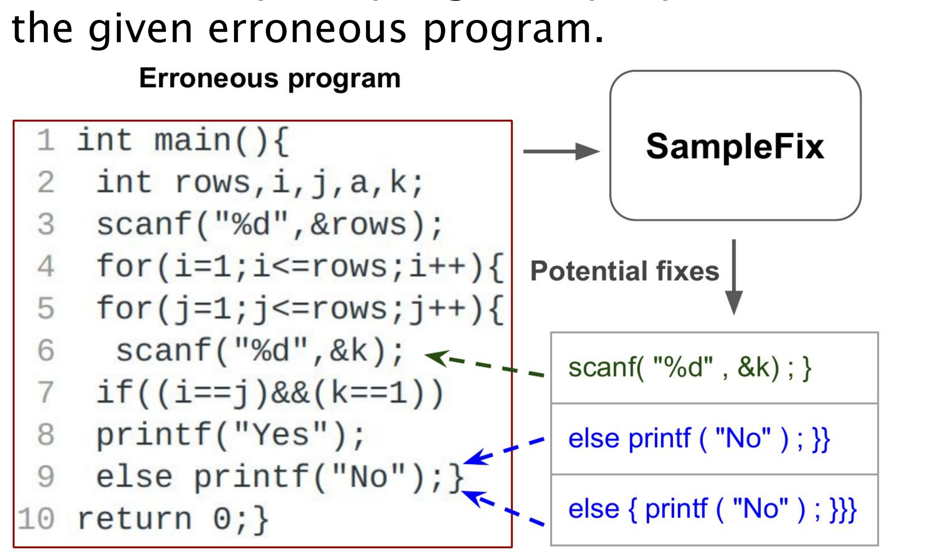
Hossein Hajipour[1], Apratim Bhattacharya[2], Mario Fritz[1]

[1]CISPA Helmholtz Center for Information Security, Germany
[2]Max Planck Institute for Informatics, Germany

Introduction

- **Automatic program correction** holds the potential of improving the productivity of programmers.
- A key challenge is ambiguity, as multiple codes can implement the same functionality.
- Therefore, we propose a **deep generative model** to automatically correct programming errors by learning a **distribution** over the fixes.
- Our evaluations on common programming errors show the **effectiveness** of the generation of **diverse fixes**.

SampleFix captures the inherent ambiguity of the possible fixes by sampling multiple potential fixes for the given erroneous program.



Contribution

- We propose an **efficient generative method** to automatically correct programming errors.
- We propose a novel regularizer to encourage the model to generate **diverse fixes**.
- Our proposed approach shows strong improvement over state-of-the-art methods.

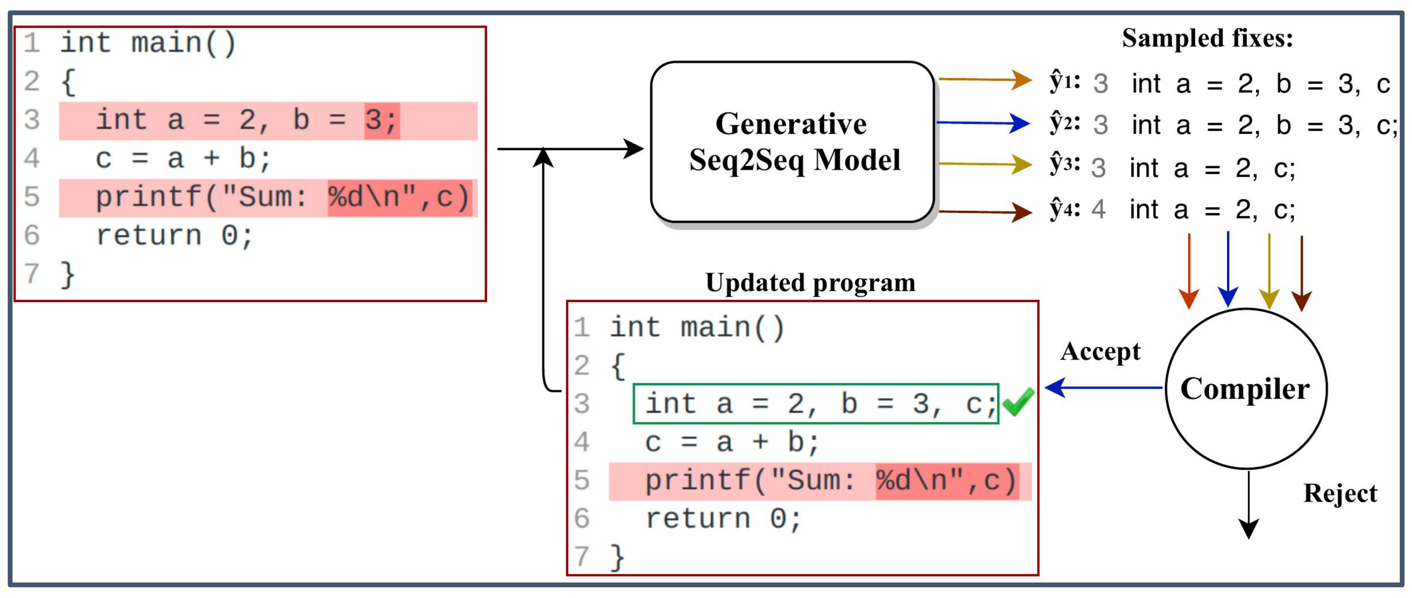
Take-aways:

- **Task:** Automatically correct common programming errors.
- **Insight:** Multiple fixes can implement the same functionality, and there is uncertainty on the intention of the programmer.
- **Our approach:** We propose a generative framework to account for inherent ambiguity and lack of representative datasets
- **Results:** Our approach resolved 65% of error messages.

SampleFix: Generative Model for Code Fixes

To resolve the programming errors:

- For a given erroneous program, the **generative model** draws **T** fixes.
- To select one out of **T** fixes, we employ a **compiler** which evaluates the fixes.
- The compiler selects the **fix** which resolves the largest number of **errors**.
- To resolve the remaining error(s), we iteratively input the updated program to the **generative model.**



Formulation:

- **Conditional Variational Autoencoders** for generating fixes.

$$\hat{\mathcal{L}}_{\text{CVAE}} = \frac{1}{T}\sum_{i=1}^{T} \log(p_\theta(\mathbf{y}|\hat{\mathbf{z}}_i, \mathbf{x})) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) \,||\, p(\mathbf{z}|\mathbf{x})) \ .$$

- Enabling diverse samples using the **Best of Many** objective (**BMS**).

$$\hat{\mathcal{L}}_{\text{BMS}} = \max_{i} \left(\log(p_\theta(\mathbf{y}|\hat{\mathbf{z}}_i, \mathbf{x})) \right) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) \,||\, p(\mathbf{z}|\mathbf{x})) \ .$$

- **DS-SampleFix**: Encouraging diversity with a diversity-sensitive regularizer.

$$\hat{\mathcal{L}}_{\text{DS-BMS}} = \max_{i} \left(\log(p_\theta(\mathbf{y}|\hat{\mathbf{z}}_i, \mathbf{x})) \right) + \boxed{\min_{i,j} d(\hat{\mathbf{y}}^i, \hat{\mathbf{y}}^j)} - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) \,||\, p(\mathbf{z}|\mathbf{x})) \ .$$

Encouraging the two closest fixes to have the maximum distance.

In the equations:

- $\mathbf{x}$: Erroneous program
- $\mathbf{y}$: Fix for the program
- $\mathbf{Z}$: Latent variable
- T : Number of samples

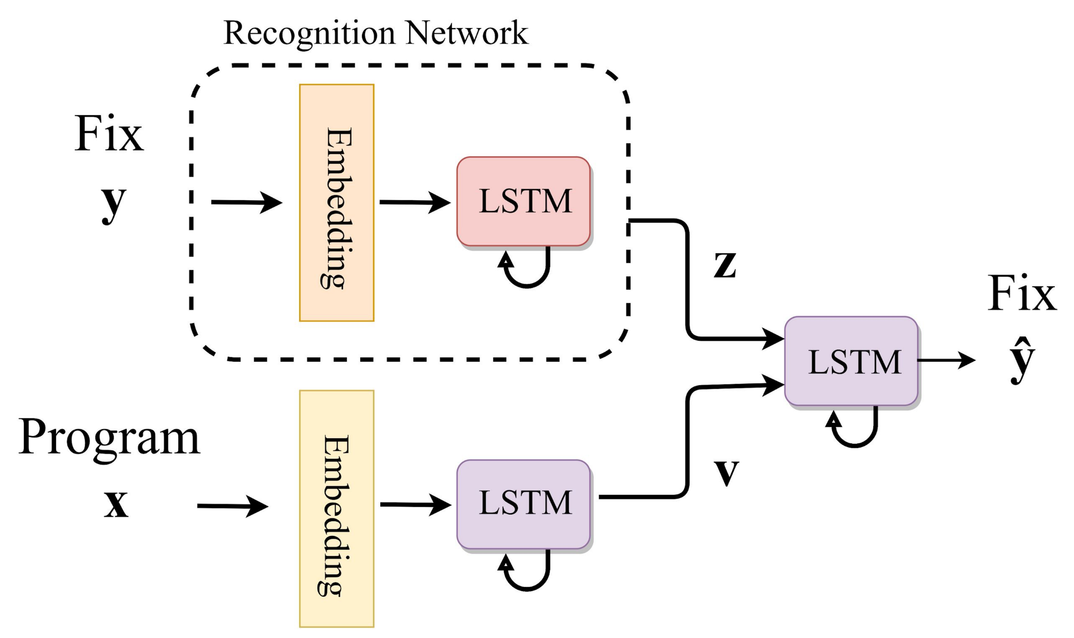
Beam search decoding:

- We employ the beam search decoding to sample more diverse fixes.
- To sample multiple fixes we decode with beam width of size **K** for each sample **z.**

Model Architecture and Implementation Details

Model architecture:

- Our generative model is based on the sequence-to-sequence architecture, similar to [1].
- All of the networks in our framework consists of 4-layers of LSTM cells with 300 units.
- The recognition network is available to encode the fixes to latent variables **Z** only during training.



Implementation details:

- we train two models, one for repairing the typo errors and another one for miss dec errors.
- We use T = 2 samples to train our models and T = 100 samples during inference time.

Experiments

Dataset:

- The **synthetic data** contains the erroneous programs which are synthesized by mutating correct programs written by students.
- The **real-world data** contains 6975 erroneous programs with 16766 error messages written by students.

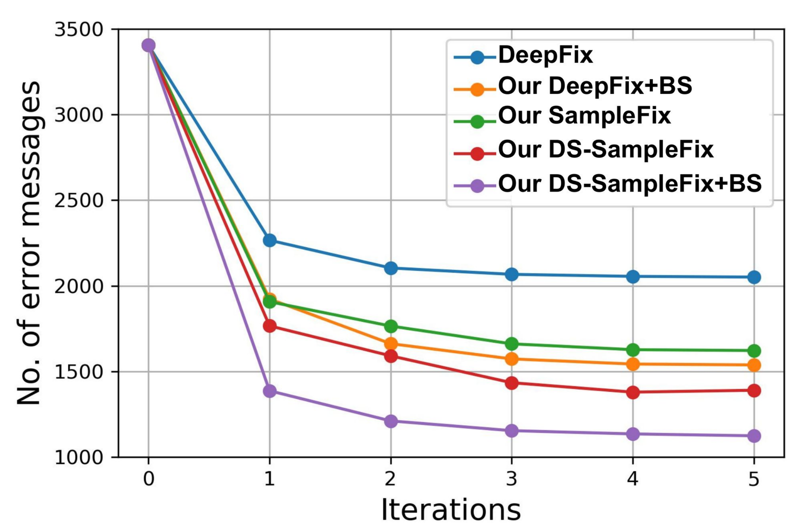
Results:

- Results of DeepFix, RLAssist, DrRepair, DeepFix + BS(beam search), SampleFix, DS-SampleFix, and DS-SampleFix + BS (beam search).
- Typo, Miss Dec, and All refer to typographic, missing variable declarations, and all of the error messages respectively.
- ✔ denotes completely fixed programs. 🐞 denotes resolved error messages.

Models	Typo		Miss Dec		All		Speed (s)
	✔	🐞	✔	🐞	✔	🐞	
DeepFix [1]	23.3%	30.8%	10.1%	12.9%	33.4%	40.8%	-
RLAssist [2]	*26.6%*	*39.7%*	-	-	-	-	-
DrRepair [3]	-	-	-	-	34.0%	-	-
Our DeepFix+ BS	25.8%	38.9%	16.8%	35.3%	39.0%	56.9%	4.82
Our SampleFix	24.8%	38.8%	16.1%	22.8%	40.9%	56.3%	0.88
Our DS-SampleFix	27.7%	40.9%	16.7%	24.7%	44.4%	61.0%	0.88
Our DS-SampleFix + BS	**27.8%**	**45.6%**	**19.2%**	**47.9%**	**45.2%**	**65.2%**	1.17

Effectiveness of Iterative Repair:

- To resolve the multiple errors in a program we use the iterative repair strategy.
- We use up to 5 iterations to resolve multiple error messages.
- We can see that after 5 iterations, SampleFix, and DS-SampleFix resolve more error messages than DeepFix.



Qualitative Example:

- Diverse fixes are generated by our DS-SampleFix. The error is highlighted at line 13.

Erroneous Program

```
1 #include <stdio.h>
2 int main (){
3 int a , i ;
4 scanf ( "%d\n" ,& a );
5 int s [ a ], p [ a ], g [ a ];
6 for ( i = 0 ; i < a ; i ++){
7 scanf ( "%d" ,& s [ i ]);}
8 for ( i = 0 ; i < a ; i ++){
9 scanf ( "%d" ,& p [ i ]);}
10 for ( i = 0 ; i < a ; i ++){
11 g [ p [ i ]]= s [ i ];}
12 for ( i = 0 ; i < a ; i ++){
13 printf ( "%d" , g [ i ]);
14 printf ( "end" );
15 return 0 ;}
```

#	DS-SampleFix Output	
1	13 printf ("%d" , g [i]); }	✔
2	9 scanf ("%d" ,& p [i]);}}	✘
3	14 printf ("end"); }	✔
4	13 printf ("%d" , g [i]); }}	✘
5	11 g [p [i]]= s [i];}}}	✘

Key Findings:

- The results show that generating multiple diverse fixes can lead to substantial improvement in the performance of the models.
- In these results, we can see CVAE and beam search decoding are complementary, while CVAE is computationally more efficient in comparison to beam search decoding.
- The performance advantage of **DS-SampleFix**, over **SampleFix** shows the effectiveness of our **novel regularizer**.

References

[1] R. R. Gupta, S. Pal, A. Kanade, and S. K. Shevade. Deepfix: Fixing common c language errors by deep learning. In AAAI, 2017.
[2] R. Gupta, A. Kanade, and S. Shevade. Deep reinforcement learning for programming language correction. In AAAI, 2019.
[3] M. Yasunaga and P. Liang. Graph-based, self-supervised program repair from diagnostic feedback. In ICML, 2020.