<DCSE> DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Ing. Jan Pašek

Master's Thesis

Computer Science and Engineering
Software Engineering
2021/2022

Supervisor:
Ing. Miloslav Konopík, Ph.D.

FACULTY OF APPLIED SCIENCES
UNIVERSITY OF WEST BOHEMIA

Source Code Generation from Descriptions in a Natural Lanuage

Abstract

This work introduces CodeFormer, a Python source code generator pretrained on a massive GitHub crawl consisting of 230M Python functions. The released model, built on BART architecture, generates Python functions based on descriptions in English. On a CodeSearchNet dataset, the CodeFormer sets a new state of the art with 46.12 BLEU, representing an improvement of 13.86 BLEU. We also release a new parallel corpus for code generation called Stack Overflow Code Generation Dataset (SOCGD), on which our model sets a baseline of 47.68 BLEU. The resulting model is ready to be integrated into a source code suggestion system in an IDE, where it can improve software developers' productivity. During our research, we discovered a better way of training the BART for machine translation. While our approach improves code generation, it must yet be verified in machine translation of natural languages.

Realized Solution

We present CodeFormer, a source code generation neural network model based on the BART architecture. First, the model is pre-trained on 230M Python functions from public GitHub repositories using a denoising objective. An example of our model's denoising capabilities can be found below.

Denosing Example noisy

```
def ion ( ) :
  [MASK] ( True [MASK]_repl ding_displayhook ()
```

denoised by model

```
def def ion():
  matplotlib.interactive(True)
  install_repl_displayhook()
```

correct

```
def ion ():
  matplotlib.interactive(True)
  install_repl_displayhook()
```

After the pre-training, we amend our model with an additional encoder to adapt the CodeFormer to a machine translation task. This setup translates natural language descriptions of functions into function source codes. In the resulting architecture, the additional encoder produces a contextual representation of a noisy translation, which is later processed and denoised by the pre-trained part of the model. An example of a source code generated by CodeFormer can be found below.

Example of a Generated Source Code

Prompt: load JSON file and parse it into a dictionary

```
import json

with open('data.json') as f:
  data = json.load(f)

print(data)
```

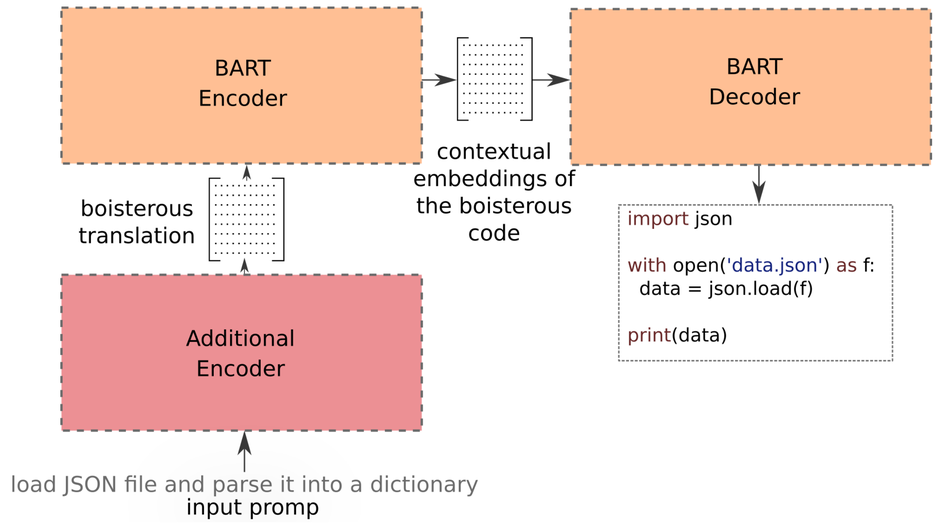
Achieved Results

The table below presents the results achieved with our model on the CodeSearchNet and compares our model to other related works. The table shows that the CodeFormer sets a new state of the art with 46.12 BLEU and 79.97% Python Validity (PV).

Model	BLEU	PV
CodeFormer + random encoder	36.35	**88.15**
CodeFormer + MQDD	39.67	71.47
CodeFormer + CodeBERT	**46.12**	79.97
REDCODER-EXT	24.43	-
GPT-2 (fine-tuned for code)	22.00	-
TranX + API knowledge	32.26	-
CodeT5	16.74	9.57

Conclusion & Future Work

This work can be followed up by further research of other applications of our pre-trained CodeFormer model, such as automated code repair or code migration. Moreover, it is possible to extend our approach by training a joint source code generator for multiple programming languages. Last but not least, we see great potential in extending our findings of preventing additional encoder erosion that occurs when using the standard BART architecture for machine translation.



Visualization of CodeFormer's architecture