跨语言克隆代码检测

成果说明

1. **功能模块说明**

代码克隆主要分为4类。广泛被认可的标准为，类型1：两段代码完全相同；类型2：除变量、类型、文字和函数名称的改动之外，两段代码其他内容均相同；类型3：两段代码相似，存在语句添加、删除、修改或代码布局的修改；类型4：两段代码实现的功能相同，但编码方式不同。目前，大多数工具仅支持类型1-3克隆的检测，无法进行类型4克隆的检测。

跨语言克隆指不同编程语言的语义相似代码，属于类型4克隆。跨语言克隆代码检测的难点在于：不同编程语言的代码具有不同的语法结构，难以识别和区分。现有方法多采用预训练模型对代码片段进行编码或基于预训练模型在下游任务上进行参数微调的方式，将代码转化为高维空间的语义向量，再通过比较这些语义向量的余弦相似度来检测跨语言克隆。此类方法的F1值通常限制在0.7至0.8之间。

本功能模块采用的方法基于“预训练-微调”的流程范式，首先使用InfoNCE损失函数对CodeBERT/GraphCodeBERT模型进行对比学习，实现将不同编程语言代码片段的语义表示在同一高维向量空间中对齐；然后使用该模型，在跨语言克隆代码检测任务上进行微调，模型可利用对齐的语义向量有效地识别出代码片段中相似的语义，具有更好的分类预测效果。

针对跨语言克隆代码数据样本不足的问题，本实验还利用数据增强技术：源代码“翻译”模型—Transcoder预训练模型，将微软公开的CodeSearchNet数据集中的Java函数“翻译”成对应的Python函数，同时，过滤掉语法不合规的Python函数，构建出Java-Python克隆函数对样本，并在预训练任务中使用。数据增强技术使得数据样本的数量和多样性大幅提升，有效地提升了模型的检测效果。

1. **环境配置**

本功能模块采用Python语言开发，使用PyTorch和CUDA框架在sec-srv8服务器(Linux Ubuntu 20.04.2)上进行训练和测试，所需各类环境的要求如下：

1. 操作系统

Linux Ubuntu 20.04.2

1. 运行及开发环境

Python 3.10.13, pip 23.3, torch 1.13.0, CUDA 11.4

其他第三方依赖见/xlccd/requirements.txt

1. 网络环境

至少10M带宽

1. 其他要求

科学上网，可访问HuggingFace，下载预训练模型

至少5G的硬盘存储空间，存储训练好的模型

1. **数据集**
2. 预训练任务数据集

数据直接来源：microsoft/UniXcoder/downstream-tasks/zero-shot-search

数据间接来源：CodeNet, two online judge websites: AIZU and AtCoder

来源链接：Zero-shot Code-to-Code Search (<https://github.com/microsoft/CodeBERT/blob/master/UniXcoder/downstream-tasks/zero-shot-search/README.md>)

数据预处理：去除注释和特殊字符，筛选出可解析成AST的solutions

数据描述：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Programming Language | Total number of problems | Total number of solutions | Average #num of solutions per problem | Average  #num of lines | Average  #num of code tokens |
| Java | 2001 | 17785 | 7.49 | 110.94 | 935.5 |
| Python | 2001 | 15399 | 7.53 | 22.93 | 209.86 |

数据集存储位置：/dataset/pre-training/Java-Python

数据集文件名称：pair\_train.jsonl (training set), pair\_valid.jsonl (validation set)

数据集划分 (based on the problem ‘task\_id’ property):

In total: 143900 clone Java-Python solution pairs

Training set: 129456 clone Java-Python solution pairs

Validation set: 14444 clone Java-Python solution pairs

1. 基于数据增强的预训练任务数据集

数据直接来源：microsoft/CodeBERT/GraphCodeBERT/codesearch

数据间接来源：CodeSearchNet, comment-code pairs from open-source libraries

来源链接：dataset.zip

(<https://github.com/microsoft/CodeBERT/tree/master/GraphCodeBERT/codesearch>)

需运行脚本下载数据：



数据预处理：筛选出可解析成AST的functions

数据描述：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Programming Language | Total number of functions (original) | Total number of functions (filtered) | Average #num of lines | Average #num of code tokens |
| Java  (original) | 181,061  (164,923/16,138) | 178,087  (162,174/15,913) | 14.26 | 137.03 |
| Python (generated) | -- | 178,087  (162,174/15,913) | 9.1 | 111.31 |

Transcoder预训练模型：/checkpoint/Transcoder/translator\_transcoder\_size\_from\_DOBF.pth

数据增强（源代码翻译）运行脚本（需自行配置transcoder的运行环境）：/xlccd/core/scripts/run\_augmentation.sh

数据增强（源代码翻译）运行代码：

/xlccd/core/transcoder/pipeline.py

数据集存储位置：/dataset/pre-training/CSN-Augmented

数据集文件名称：pair\_train.jsonl (training set), pair\_valid.jsonl (validation set)

数据集划分：

In total: 178087 clone Java-Python function pairs

Training set: 162174 clone Java-Python function pairs

Validation set: 15913 clone Java-Python function pairs

1. 微调任务数据集

数据直接来源：Chenning-Tao/C4

数据间接来源：two online coding competitions: AtCoder and Google CodeJam

来源链接：dataset

(<https://github.com/Chenning-Tao/C4/tree/main/dataset>)

数据预处理：去除注释和特殊字符，筛选出可解析成AST的solutions

数据描述：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Programming Language | Total number of problems  (train/valid/test) | Total number of solutions  (train/valid/test) | Average  #num of lines | Average  #num of code tokens |
| Java | 989  (804/97/88) | 5654  (4572/560/522) | 64.49 | 496.59 |
| Python | 989  (804/97/88) | 6211  (4994/640/577) | 20.69 | 213.93 |

数据集存储位置：/dataset/fine-tuning/Java-Python

数据集文件名称：pair\_train.jsonl (training set), pair\_valid.jsonl (validation set), pair\_test.jsonl (test set)

数据集划分 (based on the problem ‘task\_id’ property):

In total: 81726 clone and non-clone Java-Python solution pairs (40863 each)

Training set: 65314 clone and non-clone Java-Python solution pairs (32657 each)

Validation set: 8212 clone and non-clone Java-Python solution pairs (4106 each)

Test set: 8200 clone and non-clone Java-Python solution pairs (4100 each)

1. **运行脚本**
2. 基线实验

运行脚本文件名称：

训练脚本：/xlccd/core/scripts/run\_fine\_tune\_train.sh

测试脚本：/xlccd/core/scripts/run\_fine\_tune\_test.sh

脚本运行代码文件名称：

/xlccd/core/fine\_tuning\_procecure.py

运行命令（在特定的Python运行环境下）：

cd /xlccd/core/scripts

bash run\_fine\_tune\_train.sh OR bash run\_fine\_tune\_test.sh

1. C4实验 (C4 Experiment)

运行脚本文件名称：

训练脚本：/xlccd/core/scripts/run\_c4\_train.sh

测试脚本：/xlccd/core/scripts/run\_c4\_test.sh

脚本运行代码文件名称：

/xlccd/core/c4\_distributed.py

运行命令（在特定的Python运行环境下）：

cd /xlccd/core/scripts

bash run\_c4\_train.sh OR bash run\_c4\_test.sh

1. 对比实验——预训练任务

运行脚本文件名称：

/xlccd/core/scripts/run\_pre\_train.sh

脚本运行代码文件名称：

/xlccd/core/pre\_training\_procecure.py

运行命令（在特定的Python运行环境下）：

cd /xlccd/core/scripts

bash run\_pre\_train.sh

1. 对比实验——微调任务

运行脚本文件名称：

训练脚本：/xlccd/core/scripts/run\_fine\_tune\_train.sh

测试脚本：/xlccd/core/scripts/run\_fine\_tune\_test.sh

运行代码文件名称：

/xlccd/core/fine\_tuning\_procecure.py

运行命令（在特定的Python运行环境下）：

cd /xlccd/core/scripts

bash run\_fine\_tune\_train.sh OR bash run\_fine\_tune\_test.sh

1. **实验结果**
2. 基线实验

**实验内容：**使用微调任务数据集在CodeBERT/GraphCodeBERT模型上直接进行跨语言(Java-Python)克隆代码检测任务的训练和测试。

实验结果（模型及日志）：sec-srv8服务器 /dangrong目录下

/work/semantic-code-clone/result/Baseline/CodeBERT

/work/semantic-code-clone/result/Baseline/GraphCodeBERT

1. C4实验 (C4 Experiment)

**实验内容：**使用C4任务数据集(/dataset/fine-tuning/C4)在CodeBERT/GraphCodeBERT模型上进行C4跨语言(Java-Python)克隆代码检测任务的训练和测试。

实验结果（模型及日志）：sec-srv8服务器 /dangrong目录下

/work/semantic-code-clone/result/Baseline/C4/CodeBERT

/work/semantic-code-clone/result/Baseline/C4/GraphCodeBERT

1. 对比实验：预训练和微调任务

**实验内容：**基于“预训练-微调”流程范式，首先使用预训练任务数据集在CodeBERT/GraphCodeBERT模型上进行对比学习，然后使用预训练后的模型，在跨语言(Java-Python)克隆代码检测任务上进行微调，得到最终的模型。

实验结果（模型及日志）：sec-srv8服务器 /dangrong目录下

/work/semantic-code-clone/result/Experiment/fine-tune/CodeBERT

/work/semantic-code-clone/result/Experiment/fine-tune/GraphCodeBERT

1. 对比实验：基于数据增强的预训练任务和微调任务

**实验内容：**使用Transcoder模型将微软公开的CodeSearchNet数据集中的Java函数“翻译”成对应的Python函数，过滤掉语法不合规的生成结果，构建Java-Python克隆函数对预训练数据集，然后完成“预训练-微调”的全流程训练。

实验结果（模型及日志）：sec-srv8服务器 /dangrong目录下

/work/semantic-code-clone/result/Experiment/fine-tune/CSN-Augmented/CodeBERT

/work/semantic-code-clone/result/Experiment/fine-tune/CSN-Augemented/GraphCodeBERT

实验结果统计数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验类别 | 模型名称 | 准确率  Precision | 召回率  Recall | F1值 |
| 基线实验 | CodeBERT | 72.93% | 82.39% | 0.7737 |
| GraphCodeBERT | 83.75% | 82.98% | 0.8336 |
| C4实验 | CodeBERT | 91.47% | 80.24% | 0.8529  (+0.0792) |
| GraphCodeBERT | 92.58% | 78.20% | 0.8478  (+0.0142) |
| 对比实验 | CodeBERT | 96.80% | 87.68% | 0.9201  (+0.1464) |
| GraphCodeBERT | 97.15% | 85.73% | 0.9109  (+0.0773) |
| 数据增强  对比实验 | CodeBERT | 94.41% | 89.02% | 0.9164  (+0.1427) |
| GraphCodeBERT | 96.71% | 88.17% | 0.9224  (+0.0888) |