

“专业专题训练” 课程答辩

微软实习

孙子平

清华大学

2019 年 9 月 2 日

目录

1 实习任务

2 完成情况

- 数据预处理
- 编写模型
- 训练调参

3 收获

- 深度学习
- 团队合作

4 总结

实习任务介绍

实习任务

代码搜索，即用自然语言查询代码。最后以 VS Code 插件的形式交给用户。

分组

依据查询得到的代码的语言，分两组：
Python 和 JavaScript 组。

实习任务具体步骤

- ① 收集代码库（交给了队友）
- ② 建立模型
 - ① 数据预处理
 - ② 编写模型
 - ③ 训练调参
- ③ 编写服务端（部分交给了队友）
- ④ 编写客户端，即 VS Code 插件（交给了李超导师）

我的任务主要是复现 CODEnn 模型。其他人则负责复现 Baseline 和 Code2Vec 等模型。

目录

- ① 实习任务
- ② 完成情况
 - 数据预处理
 - 编写模型
 - 训练调参
- ③ 收获
 - 深度学习
 - 团队合作
- ④ 总结

目录

- 1 实习任务
- 2 完成情况
 - 数据预处理
 - 编写模型
 - 训练调参
- 3 收获
 - 深度学习
 - 团队合作
- 4 总结

数据处理步骤：

- ① 提取
- ② 清洗
- ③ 切分
- ④ 转化

注意

不同数据集和模型的预处理可能共享或者跳过一些步骤。

此外，还希望根据需求自动下载原始或处理过的数据集。

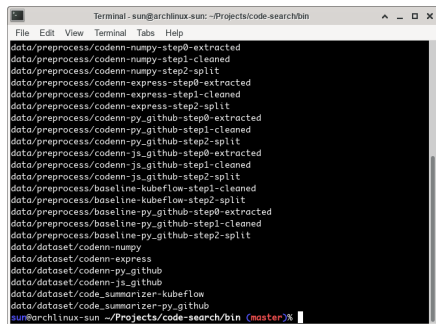
因而，整合所有的数据预处理是**复杂的**。

数据预处理设计

最初设计是使用 Makefile，而后重构，改用自己的脚本：

- `download.py`：下载并解压数据集
- `prepare.py`：按照依赖运行数据预处理脚本，生成所需数据集

下图是一部分数据集列表：



```
Terminal - sun@archlinux-sun: ~/Projects/code-search/bin
File Edit View Terminal Tabs Help
data/preprocess/codenn-numpy-step0-extracted
data/preprocess/codenn-numpy-step1-cleaned
data/preprocess/codenn-numpy-step2-split
data/preprocess/codenn-express-step0-extracted
data/preprocess/codenn-express-step1-cleaned
data/preprocess/codenn-express-step2-split
data/preprocess/codenn-py_github-step0-extracted
data/preprocess/codenn-py_github-step1-cleaned
data/preprocess/codenn-py_github-step2-split
data/preprocess/codenn-js_github-step0-extracted
data/preprocess/codenn-js_github-step1-cleaned
data/preprocess/codenn-js_github-step2-split
data/preprocess/baseline-kubeflow-step1-cleaned
data/preprocess/baseline-kubeflow-step2-split
data/preprocess/baseline-py_github-step0-extracted
data/preprocess/baseline-py_github-step1-cleaned
data/preprocess/baseline-py_github-step2-split
data/dataset/codenn-numpy
data/dataset/codenn-express
data/dataset/codenn-py_github
data/dataset/codenn-js_github
data/dataset/code_summarizer-kubeflow
data/dataset/code_summarizer-py_github
sun@archlinux-sun ~/Projects/code-search/bin (master) %
```

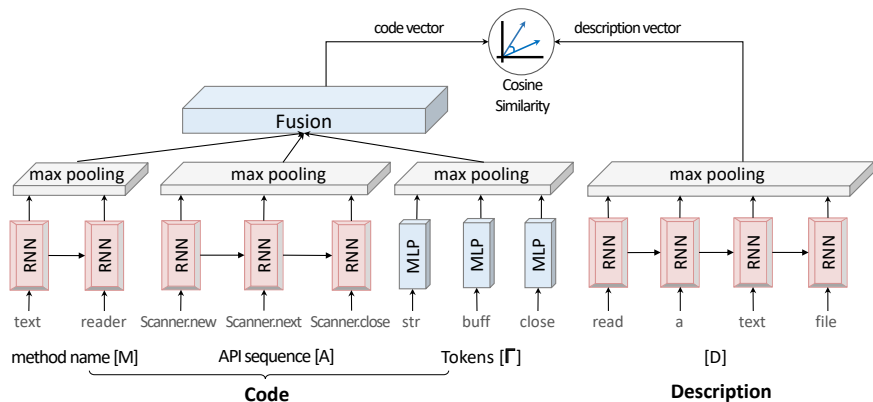
下面是数据预处理脚本列表：

```
baseline-step0-extract_py.py
baseline-step1-clean.py
baseline-step2-split.py
codenn-step0-extract_js.js
codenn-step0-extract_py.py
codenn-step1-clean.py
codenn-step2-split.py
codenn-step3-convert.py
code_summarizer-step3-convert.py
download.py
prepare.py
```


目录

- 1 实习任务
- 2 完成情况
 - 数据预处理
 - 编写模型
 - 训练调参
- 3 收获
 - 深度学习
 - 团队合作
- 4 总结

模型介绍



我重新实现了 CODEnn 的代码，从而更好地支持了多卡训练、断点、可视化，并适应了我们的数据集。

代码组织

```
bin/                               # 可执行文件所在的目录
  baseline-step0-extract_py.py ... # 数据预处理脚本
  code_search -> ../code_search    # 符号链接指向库
  codenn.py & code_summarizer.py   # 模型训练和预测脚本
  download.py & prepare.py         # 自动处理脚本
code_search/                        # 库的目录
  convert/                          # 数据处理的库
  model/                             # 模型的库
    dataset/                         # 加载数据的库
    model/                           # 最终模型的库
data/                                # 数据存放的目录
  rawcode/                          # 原始数据
  preprocess/                       # 预处理中的数据
  dataset/                          # 预处理完的数据
  models/                            # 模型训练产生的数据
LICENSE                             # MIT协议
CHANGELOG.md                        # 变更日志
README.md                           # 说明文档
package.json                        # JavaScript 依赖列表
requirements.txt                    # Python 依赖列表
```

代码组织经过重构后，有如下的改进：

- 按照标准 Python 工程结构
- 包含标准的更改记录方式（目前已经发布 0.1.0 版本）
- 支持多种数据集和多种模型
- 可执行脚本有统一的命令行参数

目录

- 1 实习任务
- 2 完成情况
 - 数据预处理
 - 编写模型
 - 训练调参
- 3 收获
 - 深度学习
 - 团队合作
- 4 总结

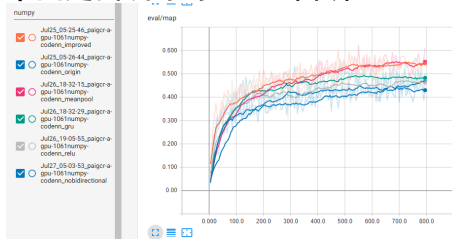
CODEnn 有很多超参可以调，如下（原作选取的用粗体显示）：

- RNN：原始 RNN、**LSTM** 或 GRU
- 池化：**Max** 或 Average
- 激活：RELU 或 **Tanh**
- Bidirectional：**是或否**
- 变长数据截断或填充的长度
- vocabulary 大小、embedding 大小、representing 大小

Numpy 数据集调参

由于时间的限制，我选取 Numpy 源代码这个小数据集供调参使用。

下图是我训练的 MAP 曲线。

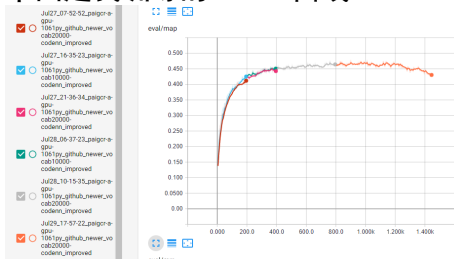


从曲线中我们能得出以下结论：

- 使用 Average 池化、GRU、RELU 激活都能对模型有所帮助
- Bidirectional 的选取很难说有没有帮助

我想知道 Vocabulary 的大小会不会影响模型性能，不得不使用大数据集。

下图是我训练的 MAP 曲线。



可以看出 Vocabulary 的大小不会影响模型性能，这说明出现频率很低的那些词确实不那么重要。得到这个结论后，我继续训练 20000 Vocabulary 的模型，发现存在性能最高点，便以 800 Epoch 作为最终模型。

模型最终性能

下表是最终模型的性能。就 $\text{acc@top } k$ 的性能指标而言，CODEnn 性能远远优于 Baseline 和 CodeNet。

dataset	acc@top 5	acc@top 10
validation set (pool=200)	0.806875	0.884375
validation set (pool=800)	0.621563	0.726250
test set (pool=200)	0.780667	0.860333
test set (pool=800)	0.596250	0.711250

没有提及的内容

有一些东西限于篇幅没有提及，在这里罗列一下：

- ① 数据预处理使用正则、nltk 和 spacy
- ② 重新实现了 ktext 库，原库已经停止维护
- ③ 尝试了更简单的模型
- ④ 集成了 Baseline 的 Code Summarizer 模型
- ⑤ 用 Tornado 编写了服务端

目录

- ① 实习任务
- ② 完成情况
 - 数据预处理
 - 编写模型
 - 训练调参
- ③ 收获
 - 深度学习
 - 团队合作
- ④ 总结

目录

- 1 实习任务
- 2 完成情况
 - 数据预处理
 - 编写模型
 - 训练调参
- 3 收获
 - 深度学习
 - 团队合作
- 4 总结

深度学习的收获

最大的收获是深度学习领域的技能有很大长进，包括 2 方面：

- PyTorch 的编程能力
- NLP 领域的理论能力

目录

- 1 实习任务
- 2 完成情况
 - 数据预处理
 - 编写模型
 - 训练调参
- 3 收获
 - 深度学习
 - 团队合作
- 4 总结

团队合作的收获

微软的团队管理很先进，实践了我在软工课上所学的知识。主要有有 3 方面：

- 冲刺管理
- PAI 分配 (GPU 服务器)
- Teams 协同

目录

- ① 实习任务
- ② 完成情况
 - 数据预处理
 - 编写模型
 - 训练调参
- ③ 收获
 - 深度学习
 - 团队合作
- ④ 总结

微软的实习让我收获了很多。但时间太过仓促，我还想尝试许多其他模型，并将它们集成进我的仓库。

谢谢大家!