ECNU

深度学习

Lab12

PostgreSQL SQL-Test Generation

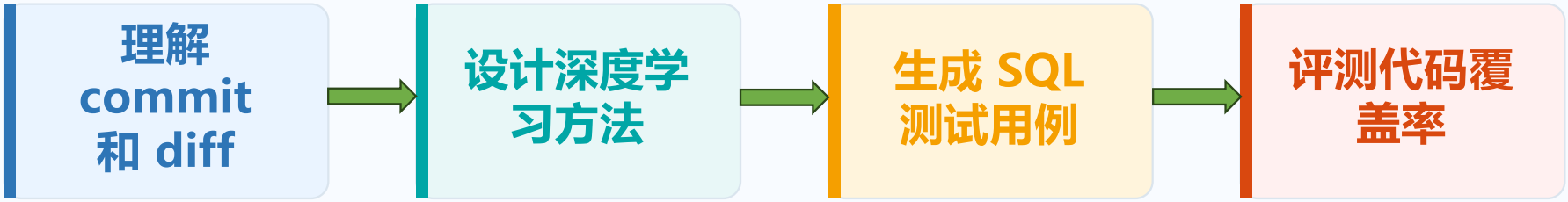
兰韵诗

助教：张仕浩 林宇轩

这是期末大作业，请在6月21号结束之前提交！

1.任务定义：PostgreSQL Commit → SQL Test

要求同学们针对 PostgreSQL 内核代码变更，通过设计 Prompt 等方式生成 SQL 测试用例，目标是尽可能覆盖 commit 中新增/修改的 PostgreSQL C 代码行



Example:

假设某个 commit 修复了一个数据库C语言源码的问题：当表里有 NULL 值时，执行 COUNT / GROUP BY 查询可能走到错误逻辑。那么我们就要生成 SQL，专门构造这个场景：

```
CREATE TABLE t (class TEXT, score INT);
INSERT INTO t VALUES
('A', 90),  ('A', NULL),  ('B', 80),  ('B', NULL);

SELECT class, COUNT(score)
FROM t
GROUP BY class;
```

代码覆盖率

代码覆盖率衡量测试用例执行时“跑到了多少源码行”。覆盖率越高，说明你的 SQL 越好地触发了 commit 新增或修改的代码逻辑。

项目结构

文件 / 目录	说明
sample.ipynb	入口模板：读取 test_v3.json，生成 output/submission.json
START_HERE.md	Quick start 先看这里，再看 README
README.md	完整项目说明、运行指南、平台评测规则
data/test_v3.json	50 条测试 commit，正式生成 SQL 的目标数据
data/train.json	100 条训练 commit，部分含参考 generated_sql_tests
run_local_eval.sh	推荐使用的一键本地评测入口
scripts/evaluate.py	统一完成格式检查、SQL 提取、coverage 指标计算、平台评测
scripts/evaluate_coverage.sh	编译 PostgreSQL、执行 SQL、生成 coverage 报告
scripts/generate_submission.py	调用 ChatECNU API 生成 submission 的参考脚本
postgresql-13.23.tar.bz2	PostgreSQL 13.23 源码包

2.Dataset

测试集 data/test_v3.json

- 共 50 个 PostgreSQL commit 记录
- 每条记录包含：
  - `id`：唯一标识，提交 JSON 中必须使用相同 id
  - `subject`：commit 标题
  - `email_body`：commit 描述
  - `patches`：代码 diff，包括 `raw_diff` 和 `diff_blocks`
  - `match_info`：代码行级匹配信息，用于和 coverage 报告对齐

训练集 data/train.json

共 100 条记录，结构与测试集类似。
额外包含 generated_sql_tests 字段。
可作为 Prompt 示例、规则参考或 SFT 训练样本。

不要直接把训练样例当成测试提交！

单条commit结构

JSON 主要字段

```
{
  "id": ... ,
  "subject": "commit title",
  "email_body": "commit message",
  "patches": [ ... ],
  "match_info": [ ... ]
}
```

id 用于提交对齐；
match_info 用于覆盖率评测

patches 主要字段

```
{
  "file": "src/backend/.../xxx.c",
  "function": "modified_function",
  "before_code": "...",
  "after_code": "...",
  "raw_diff": "...",
  "diff_blocks": [ ... ]
}
```

重点看 raw_diff / diff_blocks 中
新增或修改的逻辑

3.TO DO

1 阅读 commit

理解 subject、email_body、patches、raw_diff等关键信息，判断修改逻辑需要什么 SQL 场景触发。

2 生成 SQL 整理提交格式

用深度学习相关方法为每条测试 commit 生成 SQL 测试用例，并整理为规定的 json 格式，做好本地检查。

3 上传平台

提交 submission.json，平台自动提取 SQL、执行测试、统计覆盖率并给出评分。

baseline pipeline - scripts/generate_submission.py

其中调用了ChatECNU大模型API，可通过个人API-KEY使用，免费（注意有限额）

ChatECNU 个人API-KEY获取链接：https://developer.ecnu.edu.cn/vitepress/llm/authorization.html

该网站还有详细的ChatECNU使用方式介绍。

提交格式：submission.json

平台会提取 <sql>...</sql> 标签中的 SQL 执行

```
[
  {
    "id": 101,
    "generated_sql_tests": "<test_cases>\n  <test_case id=\"1\">\n    <description>测试 xxx 代码路径</description>\n    <sql>\n-- Setup\nDROP TABLE IF EXISTS test_t1 CASCADE;\nCREATE TABLE test_t1 (id INT);\nINSERT INTO test_t1 VALUES (1);\n\n-- Execution\nSELECT * FROM test_t1 WHERE id = 1;\n\n-- Teardown\nDROP TABLE IF EXISTS test_t1 CASCADE;\n    </sql>\n  </test_case>\n</test_cases>"
  }
]
```

格式硬性要求

- 顶层必须是 JSON 数组。
- 每个元素必须包含 id 和 generated_sql_tests。
- id 必须与 data/test_v3.json 对应。
- 用 <test_cases> 包裹测试用例。
- 每个 test case 建议包含 description 和 sql。
- SQL 写在 <sql>...</sql> 标签内。

格式错误会导致无法提取 SQL！这类问题应在上传前运行 scripts/evaluate.py check outputs/submission.json提前发现 JSON 结构、id、SQL 标签等问题并解决！

SQL避免死锁、无限递归和超长执行时间！

4.Evaluation- 本地快速检查与评测

1. 快速检查 submission

```
python3 -m pip install -r requirements.txt

python3 scripts/evaluate.py check \
outputs/submission.json \
--dataset data/test_v3.json
```

不编译 PostgreSQL，只检查 JSON 格式和 SQL case 数

2. 一键本地评测

```
# from submission JSON
./run_local_eval.sh \
  --submission outputs/submission.json \
  --name my_submission

# 已编译过可跳过 build
./run_local_eval.sh \
  --submission outputs/submission.json \
  --name my_submission \
  --skip-build
```

首次运行会解压/编译 PostgreSQL 13.23，耗时较长。

如果希望低于某个本地分数直接失败，可使用 --min-score 参数。

查看本地评测结果

结果目录

```
outputs/local_eval/<name>/
  summary.txt
  eval_result.json
  coverage_workspace/report/index.html
  coverage_workspace/psql_output.log
  coverage_workspace/run.log
```

- summary.txt：快速查看主指标
- eval_result.json：完整评测结果
- report/index.html：HTML 覆盖率报告
- psql_output.log：定位 SQL 报错

常用调试命令

```
cat outputs/local_eval/my_submission/summary.txt

grep -n "ERROR:" \

outputs/local_eval/my_submission/coverage_workspace/psql_output.log | head -50

python3 - <<'PY'
import json
with
open("outputs/local_eval/my_submission/eval_result.json")
as f:
    print(json.load(f)["summary"])
PY
```

优先处理：大量 SQL 报错、没有提取到 SQL、目标覆盖率过低。

本地评测只用于上传前自查；平台环境可能不同，最终排名和分数以打榜平台为准。

核心评测指标

核心指标	含义 / 公式	说明
PrecNF	total_covered / (total_matched - total_not_found)	主指标，越大越好
efficiency	(sql_count ^ 0.35) / (PrecNF × 100)	同分时看 SQL 使用效率，越小越好

1 PrecNF

衡量 SQL 是否真正覆盖 commit 中新增/修改的 C 代码行。
越高越好，是平台评分主要指标。

2 efficiency

衡量 SQL 使用效率。在覆盖率接近时，SQL 越精简、无效语句越少越好。

Tips：先提高 PrecNF；再减少无效 SQL和重复 workload。

线上评测

平台排序配置为：

指标	排序
`PrecNF`	降序，越大越好
`efficiency`	升序，越小越好

其中 `PrecNF` 对应评测脚本里的 `global_precision_excl_not_found` 。

`efficiency` 是用于同 `PrecNF` 情况下比较 SQL 使用效率的代价值，计算方式为：

```
efficiency = (sql_count ^ 0.35) / (PrecNF * 100)
```

其中 `sql_count` 统计的是 `<sql>...</sql>` 中的实际 SQL 语句数，而不是 `<sql>` 标签数量。计数时按顶层分号拆分，注释、字符串、quoted identifier 和 PostgreSQL dollar-quoted 函数体里的分号不会被当作新语句。 `sql_count` 只用于计算 `efficiency` ，不作为平台的单独排序指标。评测失败时，脚本会给 `PrecNF` 写 `0.000000` ，给 `efficiency` 写一个较大的默认值。

Evaluation平台提交链接： https://pg-leaderboard-deeplearning.loca.lt/leaderboard

线上评测

Warning：若发现雷同测试用例，严肃处理！

学号 | 姓名 | 昵称 | submission.json | 生成时间：2026-05-28T15:21:47+08:00

Choose File No file chosen | 提交 | CSV

RANK	NICKNAME	PRECNF	EFFICIENCY	BEST_SUBMIT_TIME	TOTAL	LAST_STATUS
1	zshyoo	0.2071	0.1510	2026-05-28T13:47:09+08:00	6	succeeded
2	lyxtest	0.2071	0.1510	2026-05-28T14:13:14+08:00	2	succeeded

提交历史：zshyoo 关闭

SUBMIT_TIME	STATUS	PRECNF	EFFICIENCY
2026-05-28T13:47:09+08:00	succeeded	0.2071	0.1510
2026-05-28T13:37:36+08:00	succeeded	0.2071	0.1510
2026-05-28T12:58:04+08:00	succeeded	0.2071	0.1510
2026-05-28T11:16:53+08:00	succeeded	0.2071	0.1510
2026-05-28T11:14:16+08:00	failed		
2026-05-28T10:20:59+08:00	failed		

5.Possible Improvement

Prompt Engineering

可参考 scripts/generate_submission.py 中的 prompt 模板

SFT 微调

可使用 data/train.json 中的 generated_sql_tests作为监督微调样本。

检索增强

混合方案

不限于上述方法，只是提供参考，大家可以自行选择高效的方法，提高覆盖率效果

6.大作业提交清单

1 平台提交 submission.json

每日每人最多提交**五次**（不含提交失败）每次取最高

+

2 方法相关代码

仅需提交生成SQL测试用例相关代码！数据和check脚本等与方法无关文件不需要提交

+

3 实验报告

一页Word内容即可

平台榜单的分值会作为主要打分依据，但我们还会考虑代码的规范程度、方案创新程度、代码工作量等对分数进行上下浮动调整。

Evaluation平台提交链接： https://pg-leaderboard-deeplearning.loca.lt/leaderboard
代码与报告提交链接：

常见问题 - 更多本地评测环境配置、调试等相关细节详见README

ECNU

SQL 报错会影响分数吗?

报错 SQL 通常不贡献覆盖率，也会浪费 case 和执行时间；建议尽量减少。

本地分数和平台不一致怎么办?

正常现象。编译器、gcov/lcov、系统环境、路径和 locale 都可能影响 coverage 报告。

本地应该看哪个指标?

重点看 summary.global_precision_excl_not_found / PrecNF，同时检查 psql_output.log 中的 ERROR。

为什么默认关闭 branch coverage?

本任务主要看行级覆盖。macOS 或 LLVM gcov 环境下，branch coverage 容易触发 lcov/genhtml 兼容问题，所以本地默认关闭更稳定。

PostgreSQL socket 路径过长怎么办?

脚本默认使用 /tmp/pgcov-55432 作为 socket 目录，避免项目路径过长导致 socket 创建失败。

端口冲突怎么办?

一键脚本可指定端口：
./run_local_eval.sh --submission outputs/submission.json --name my_submission --port 55433