

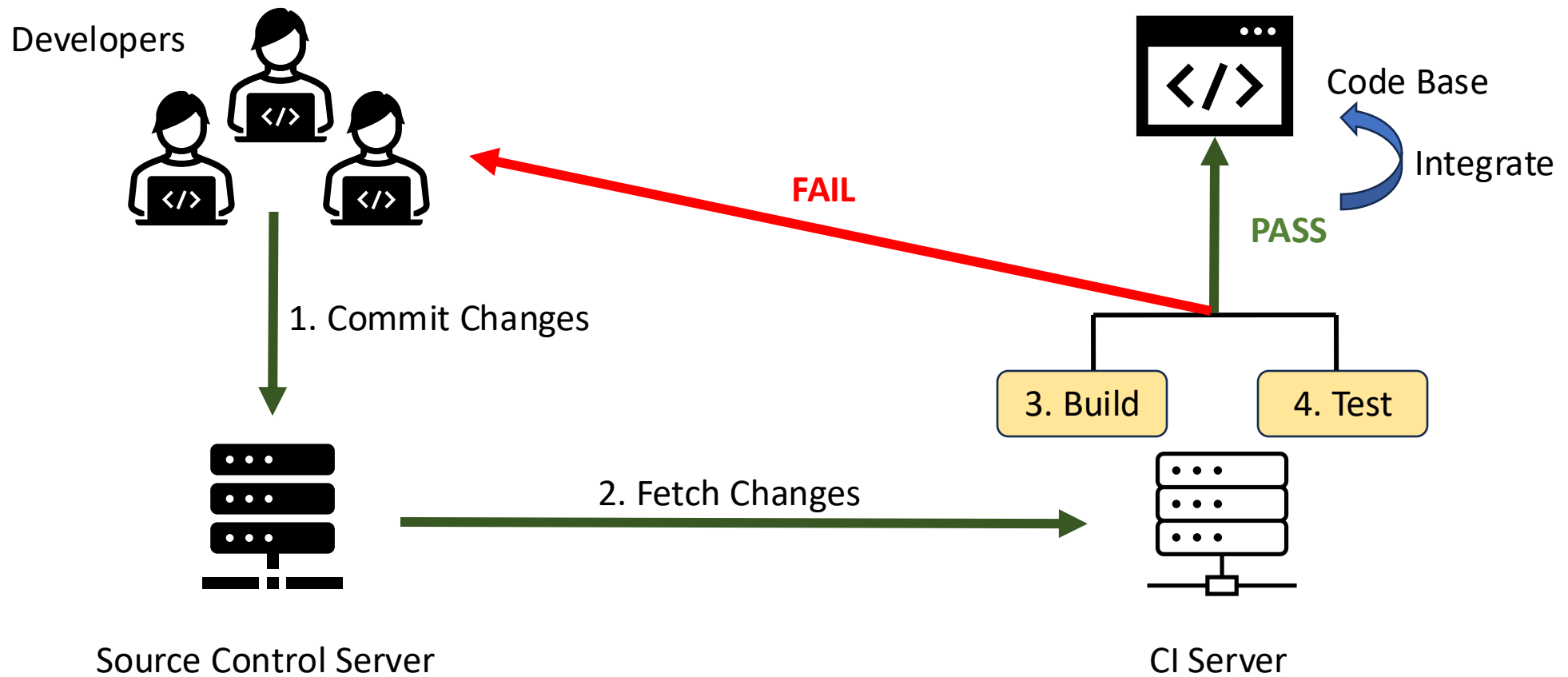
CS202: Software Tools and Techniques for CSE

Lecture 2

Shouvick Mondal

shouvick.mondal@iitgn.ac.in
August 2025

Continuous Integration (CI)



Setting up CI for a GitHub project (cs434)

The screenshot shows the GitHub repository page for 'SET-IITGN / cs434'. The repository is public and has 1 branch (main) and 0 tags. The file list shows the following files and their commit history:

File	Commit Message	Time
.circleci	Update config.yml	2 days ago
.github/workflows	Create pylint.yml	2 days ago
README.md	Create README.md	2 days ago
src.py	fixed assert	2 days ago

The 'src.py' file is highlighted with a red box. Below the file list is the README section, which contains the text 'cs434'. On the right side of the page, there are sections for 'About', 'Releases', 'Packages', and 'Languages'. The 'Languages' section shows that the repository is 100% Python.

Specify CI workflows/jobs in YAML (config.yml)

The screenshot shows the GitHub repository page for SET-IITGN/cs434. The file `.circleci/config.yml` is open, showing a CircleCI configuration. The file is 26 lines long (24 loc) and 899 Bytes. The configuration is as follows:

```
1 # Use the latest 2.1 version of CircleCI pipeline process engine.
2 # See: https://circleci.com/docs/configuration-reference
3 version: 2.1
4
5 # Define a job to be invoked later in a workflow.
6 # See: https://circleci.com/docs/configuration-reference/#jobs
7 jobs:
8   say-hello:
9     # Specify the execution environment. You can specify an image from Docker Hub or use one of our convenience images from CircleCI's Developer Hub.
10    # See: https://circleci.com/docs/configuration-reference/#executor-job
11    docker:
12      - image: cimg/base:stable
13    # Add steps to the job
14    # See: https://circleci.com/docs/configuration-reference/#steps
15    steps:
16      - checkout
17      - run:
18        name: "Check"
19        command: "python3 src.py"
20
21 # Orchestrate jobs using workflows
22 # See: https://circleci.com/docs/configuration-reference/#workflows
23 workflows:
24   say-hello-workflow:
25     jobs:
26       - say-hello
```

The file is highlighted with red boxes to show its structure:

- Lines 7-10: `jobs:` section
- Lines 11-13: `docker:` section
- Lines 14-19: `steps:` section
- Lines 23-26: `workflows:` section

The file is named `src.py` and is located in the `src` directory. The repository is named `cs434` and is public. The repository is owned by SET-IITGN. The repository is named `cs434` and is public. The repository is owned by SET-IITGN. The repository is named `cs434` and is public. The repository is owned by SET-IITGN.

Specify CI workflows/jobs in YAML (config.yml)

The screenshot displays a GitHub repository named 'SET-IITGN / cs434'. The file explorer on the left shows the repository structure, including a file named 'src.py' which is highlighted with a red box. The main area shows the 'config.yml' file for CircleCI, also with a red box highlighting the 'jobs' and 'workflows' sections. The workflow 'say-hello' is defined with a 'docker' executor and a 'check' step. A blue arrow points from the 'docker' section of the YAML to the 'Resources' tab of the workflow run, which shows it is using 'Docker / Large' executor. A green arrow points from the 'command: "python3 src.py"' line in the 'check' step to the 'Check' step in the workflow run details, which shows the command being executed and the exit code 0.

```
1 # Use the latest 2.1 version of CircleCI pipeline process engine.
2 # See: https://circleci.com/docs/configuration-reference
3 version: 2.1
4
5 # Define a job to be invoked later in a workflow.
6 # See: https://circleci.com/docs/configuration-reference/#jobs
7 jobs:
8   say-hello:
9     # Specify the execution environment. You can specify an image from
10    # See: https://circleci.com/docs/configuration-reference/#executor
11    docker:
12      - image: cimg/base:stable
13    # Add steps to the job
14    # See: https://circleci.com/docs/configuration-reference/#steps
15    steps:
16      - checkout
17      - run:
18        name: "Check"
19        command: "python3 src.py"
20
21 # Orchestrate jobs using workflows
22 # See: https://circleci.com/docs/configuration-reference/#workflows
23 workflows:
24   say-hello-workflow:
25     jobs:
26       - say-hello
```

say-hello Success

Duration / Finished: 14s / 2d ago | Queued: 0s | Executor / Resource Class: Docker / Large

STEPS TESTS TIMING ARTIFACTS RESOURCES

Parallel runs

0 00:13 Use parallelism to run faster tests
Parallelism speeds up tests by splitting them across multiple executors.

Accessible step output ☒

- ✓ Spin up environment
- ✓ Preparing environment variables
- ✓ Checkout code
- ✓ Check

Check

```
1 #!/bin/bash -eo pipefail
2 python3 src.py
3
4 CircleCI received exit code 0
```

Python 100.0%

Source code *diff* (unified)

github.com/SET-IITGN/cs202/commit/45a77e06f0b1b13907fcca80e7debd614fe0ba3f?diff=unified#diff-5cc5f480c15364f8b27b8941ae1e9498d3170d3a6e13421e7d527e7e1b5e5410

Filter files... src.c vuln.c

2 files changed +1 -2 lines changed

Search within code

src.c

```
@@ -12,7 +12,7 @@ int main() {
12 12 char *buffer;
13 13 char message[] = "Hello, worlcome to CS202";
14 14
15 - int numBytes = strlen(message);
15 + int numBytes = strlen(message);
16 16 buffer=(char*)malloc(numBytes+1);
17 17 memset(buffer, '\0', numBytes+1);
18 18 copyMemory(buffer, message, numBytes);
```

vuln.c

```
@@ -1,4 +1,3 @@
1 - //base file
2 1 #include <stdio.h>
3 2
4 3 void copyMemory(char* destination, const char* source, int numBytes) {
```

Source code *diff* (split)

Browsing a commit on GitHub

Commit 45a77e0

shouvick149 committed 2 weeks ago · 4 / 5

A few more minor edits.

main

1 parent 505adce commit 45a77e0

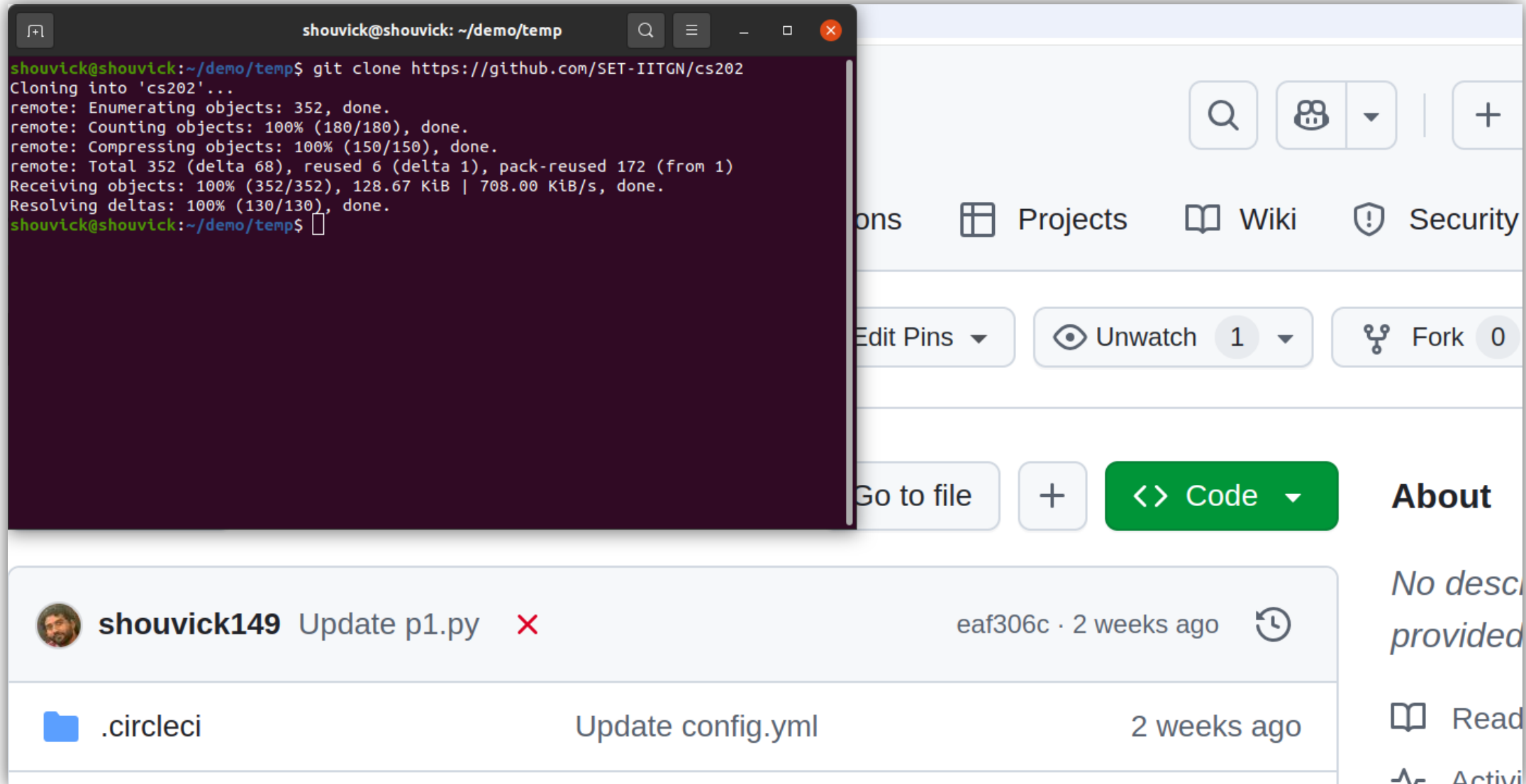
2 files changed
+1 -2 lines changed

src.c

vuln.c


```
@@ -12,7 +12,7 @@ int main() {  
12 char *buffer;
```



git clone https://github.com/SET-IITGN/cs202



The image shows a terminal window in the foreground and a GitHub repository page in the background. The terminal window, titled 'shouvick@shouvick: ~/demo/temp', displays the output of the command 'git clone https://github.com/SET-IITGN/cs202'. The output shows the cloning process, including enumerating, counting, and compressing objects, and receiving and resolving deltas. The terminal window is partially overlapping the GitHub page, which shows the repository 'shouvick149' with a commit 'eaf306c' from 2 weeks ago. The repository page also shows a file '.circleci' with a commit 'Update config.yml' from 2 weeks ago. The GitHub page includes navigation links like 'Projects', 'Wiki', and 'Security', and a 'Code' button.


```
shouvick@shouvick:~/demo/temp$ git clone https://github.com/SET-IITGN/cs202
Cloning into 'cs202'...
remote: Enumerating objects: 352, done.
remote: Counting objects: 100% (180/180), done.
remote: Compressing objects: 100% (150/150), done.
remote: Total 352 (delta 68), reused 6 (delta 1), pack-reused 172 (from 1)
Receiving objects: 100% (352/352), 128.67 KiB | 708.00 KiB/s, done.
Resolving deltas: 100% (130/130), done.
shouvick@shouvick:~/demo/temp$
```

shouvick149 Update p1.py ✖ eaf306c · 2 weeks ago 

 .circleci Update config.yml 2 weeks ago

Go to file + <> Code About

No description provided

Readme  Activity

cd cs202; git diff --help

```
shouvick@shouvick: ~/demo/temp
GIT-DIFF(1)                               Git Manual                               GIT-DIFF(1)

NAME
    git-diff - Show changes between commits, commit and working tree, etc

SYNOPSIS
    git diff [<options>] [<commit>] [--] [<path>...]
    git diff [<options>] --cached [<commit>] [--] [<path>...]
    git diff [<options>] <commit> <commit> [--] [<path>...]
    git diff [<options>] <blob> <blob>
    git diff [<options>] --no-index [--] <path> <path>

DESCRIPTION
    Show changes between the working tree and the index or a tree, changes between the index
    and a tree, changes between two trees, changes between two blob objects, or changes
    between two files on disk.

    git diff [<options>] [--] [<path>...]
    This form is to view the changes you made relative to the index (staging area for the
    next commit). In other words, the differences are what you could tell Git to further
    add to the index but you still haven't. You can stage these changes by using git-
    add(1).

    git diff [<options>] --no-index [--] <path> <path>
```

Source code *diff* (unified)

The screenshot shows a GitHub diff interface for a commit. The left sidebar lists files: `src.c` and `vuln.c`. The main area displays the diff for these files. `src.c` has 2 files changed with +1 line and -2 lines. `vuln.c` has -1 line changed. The diff shows a change in `src.c` where a line is removed and a new line is added, both using `strlen`. The `vuln.c` diff shows a removed line and a new line, both using `copyMemory`.

github.com/SET-IITGN/cs202/commit/45a77e06f0b1b13907fcca80e7debd614fe0ba3f?diff=unified#diff-5cc5f480c15364f8b27b8941ae1e9498d3170d3a6e13421e7d527e7e1b5e5410

Filter files...

src.c

vuln.c

2 files changed +1 -2 lines changed

Search within code

src.c

+1 -1

```
@@ -12,7 +12,7 @@ int main() {
12 12     char *buffer;
13 13     char message[] = "Hello, worlcome to CS202";
14 14
15 -     int numBytes = strlen(message);
15 +     int numBytes = strlen(message);
16 16     buffer=(char*)malloc(numBytes+1);
17 17     memset(buffer, '\0', numBytes+1);
18 18     copyMemory(buffer, message, numBytes);
```

vuln.c

-1

```
@@ -1,4 +1,3 @@
1 - //base file
2 1     #include <stdio.h>
3 2
4 3     void copyMemory(char* destination, const char* source, int numBytes) {
```

git diff <parent_commit> <curr_commit>

```
diff --git a/src.c b/src.c
index 0d6a029..85bb4fb 100644
--- a/src.c
+++ b/src.c
@@ -12,7 +12,7 @@ int main() {
     char *buffer;
     char message[] = "Hello, worlcome to CS202";

-    int numBytes = strlen(message);
+    int numBytes = strlen(message);
     buffer=(char*)malloc(numBytes+1);
     memset(buffer, '\0', numBytes+1);
     copyMemory(buffer, message, numBytes);
diff --git a/vuln.c b/vuln.c
index 838cba5..c1a0e33 100644
--- a/vuln.c
+++ b/vuln.c
@@ -1,4 +1,3 @@
-//base file
#include <stdio.h>
```

My Group's Contribution to the OSS community

Minecraft: Automated Mining of Software Bug Fixes with Precise Code Context

Sai Krishna Avula
Computer Science and Engineering
IIT Gandhinagar, Gujarat, India
avulasai.krishna@iitgn.ac.in

Venkatesh Vobbliseti
Metallurgical and Materials Engineering
NIT Raipur, Chhattisgarh, India
venkateshseti1211@gmail.com

Shourvik Mondal*
Computer Science and Engineering
IIT Gandhinagar, Gujarat, India
shourvik.mondal@iitgn.ac.in

MineCPP: Mining Bug Fix Pairs and Their Structures

Sai Krishna Avula
IIT Gandhinagar
Gandhinagar, India
avulasai.krishna@iitgn.ac.in

Shourvik Mondal
IIT Gandhinagar
Gandhinagar, India
shourvik.mondal@iitgn.ac.in

program-repair.org

Community-driven effort to facilitate discovery, access and systematization of data related to automated program repair research

Our dataset has been included as a multilingual program repair benchmark in the area of Automated Program Repair (APR).

Bibliography

[AutoCodeRover: Autonomous Program Improvement](#)
Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, Abhik Roychoudhury
ISSTA 2024

[Benchmarking Automated Program Repair: An Extensive Study on Both Real-World and Artificial Bugs](#)
Yicheng Ouyang, Jun Yang, Lingming Zhang
ISSTA 2024

[Automated Program Repair via Conversation: Fixing 162 out of 337 Bugs for \\$0.42 Each using ChatGPT](#)
Chunqiu Steven Xia, Lingming Zhang
ISSTA 2024

[BRAFAAR: Bidirectional Refactoring, Alignment, Fault Localization, and Repair for Programming Assignments](#)
Linna Xie, Chongmin Li, Yu Pei, Tian Zhang, Minxue Pan
ISSTA 2024

[CREF: An LLM-Based Conversational Software Repair Framework for Programming Tutors](#)
Boyang Yang, Haoye Tian, Weiguo Pian, Haoran Yu, Haitao Wang, Jacques Klein, Tegawendé F. Bissyandé, Shunfu Jin
ISSTA 2024

[ThinkRepair: Self-Directed Automated Program Repair](#)
Xin Yin, Chao Ni, Shaohua Wang, Zhenhao Li, Limin Zeng, Xiaohu Yang
ISSTA 2024

[View all »](#)

Tools

[ExtractFix](#) — repairs program vulnerabilities via crash constraint extraction

[Gin](#) — a tool for experimentation with GI

[PyGGI](#) — a Python general framework for genetic improvement

[View all »](#)

Benchmarks

[TutorCode](#) — 1,239 C++ buggy codes incorporating human tutor guidance and solution descriptions, accessible via API

[Minecraft](#) — a benchmark with C/C++, Java, and Python programs constructed via automated mining of software bug fixes with precise code context

[BugsPHP](#) — a dataset for automated program repair in PHP

[View all »](#)

Pages

[Defects4J Dissection](#) — presents data to help researchers and practitioners to better understand the Defects4J bug dataset

[RepairThemAll experiment](#) — presents experimental data obtained using RepairThemAll framework

[View all »](#)

Benchmarks

[TutorCode](#) — 1,239 C++ buggy codes incorporating human tutor guidance and solution descriptions, accessible via API

[Minecraft](#) — a benchmark with C/C++, Java, and Python programs constructed via automated mining of software bug fixes with precise code context

[BugsPHP](#) — a dataset for automated program repair in PHP

[View all »](#)

My Group's Contribution to the OSS community

Publicly Available Virtual Machine (sandbox) from SET Group – IITGN.

Provides a playground without worrying about the consequence if anything breaks!

The screenshot shows a Zenodo record page for a Virtual Machine. The browser address bar shows the URL `zenodo.org/records/10467159`. The Zenodo header includes a search bar, navigation links for 'Communities' and 'My dashboard', and buttons for 'Log in' and 'Sign up'. A notice states that Zenodo's user support line is staffed on regular business days between Dec 23 and Jan 5. The record is titled 'Virtual Machine for Software Engineering and Testing Group - IIT Gandhinagar' and was published on January 7, 2024, as version v1. It is categorized as 'Software' and is 'Open'. The author is listed as 'Mondal, Shouvick (Supervisor)'. The affiliations section lists '1. IIT Gandhinagar'. The 'System (host) requirements' section lists hardware and software requirements. The statistics sidebar shows 170 views and 75 downloads. The versions section shows 'Version v1' published on Jan 7, 2024.

zenodo Search records... Communities My dashboard Log in Sign up

Info: Zenodo's user support line is staffed on regular business days between Dec 23 and Jan 5. Response times may be slightly longer than normal.

Published January 7, 2024 | Version v1 Software Open

Virtual Machine for Software Engineering and Testing Group - IIT Gandhinagar

Mondal, Shouvick (Supervisor)¹ Hide affiliations

1. IIT Gandhinagar

System (host) requirements:

Hardware (minimum)

- RAM: 8GB
- #CPUs: 8
- Disk space: 80 GB

Software (minimum)

- Oracle VirtualBox (preferably version 6.1.30)
- Host OS: Windows/Linux/Mac OS

Statistics

	All versions	This version
Views	170	170
Downloads	75	75
Data volume	2.0 TB	2.0 TB

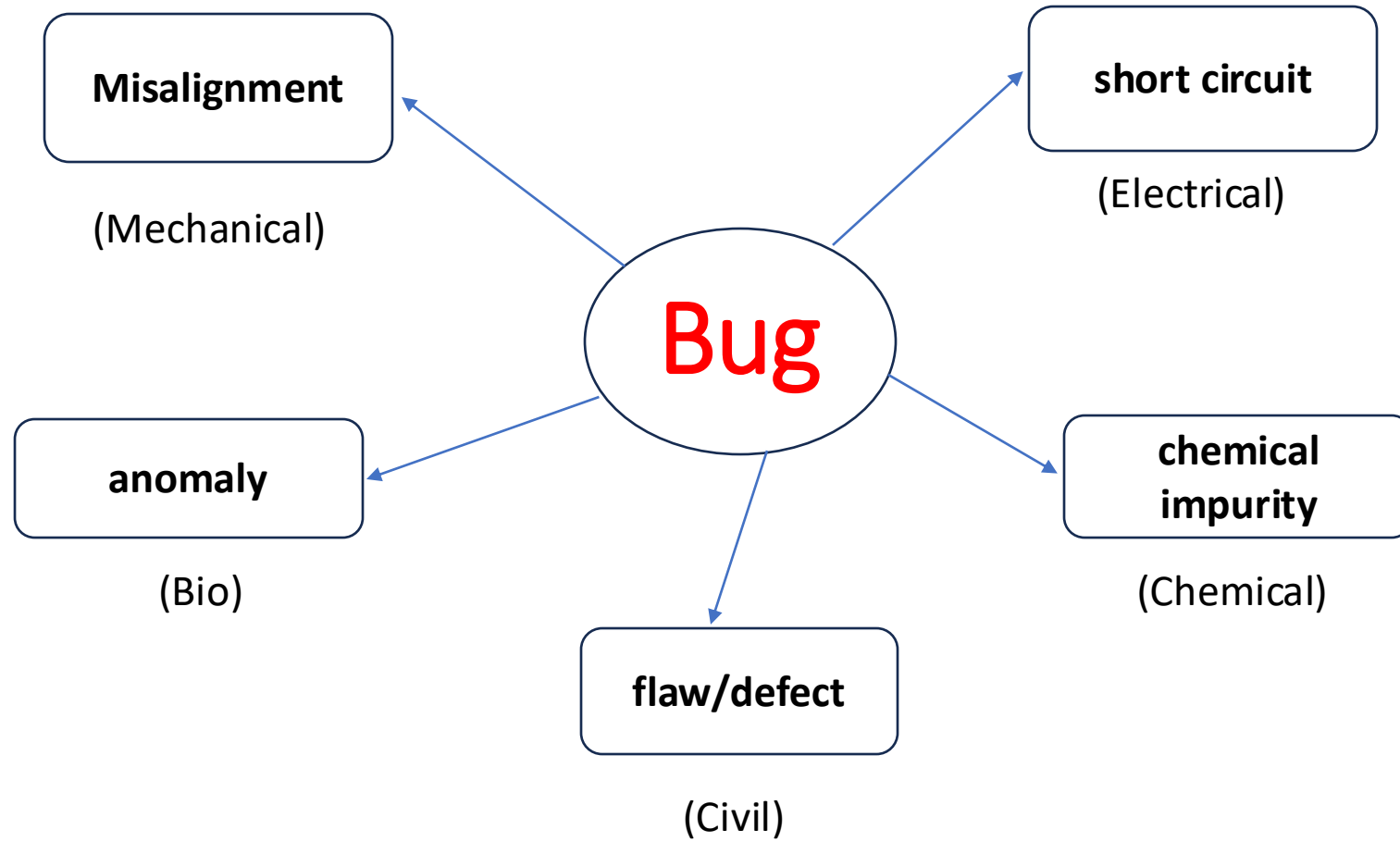
[More info on how stats are collected...](#)

Versions

Version	Published
Version v1	Jan 7, 2024

10.5281/zenodo.10467159

Analogy of Bug



Bug Mining

- Bug mining (bug extraction) – identifying, **extracting code snippets** with **bugs** along with their **corresponding fixes** and **contextual information** from software repositories.
- **Identification** and **resolution** of bugs - critical for **software reliability, performance, and security**.
- Essential for improving software quality, enhancing developer productivity, and advancing our understanding of software defects

The challenges of extracting bug and its contextual information

- **Granularity** of repository-mined bug-fixing datasets/bug-mining techniques
 - **Function level**, **without** precise bug location and bug types.
- **No** language agnostic algorithm to extract **code snippets**
- *Major issue*: bug-fix **context** has never been explored...

- **No unified** tools - bug **contextual information**
- **Unavailability** of GUI to **visualise** and **analyse** the bugs

Minecraft: Automated Mining of Software Bug Fixes with Precise Code Context

Published in ASE 2023 (CORE A*) Industry challenge track

<https://www.doi.org/10.1109/ASE56229.2023.00116>

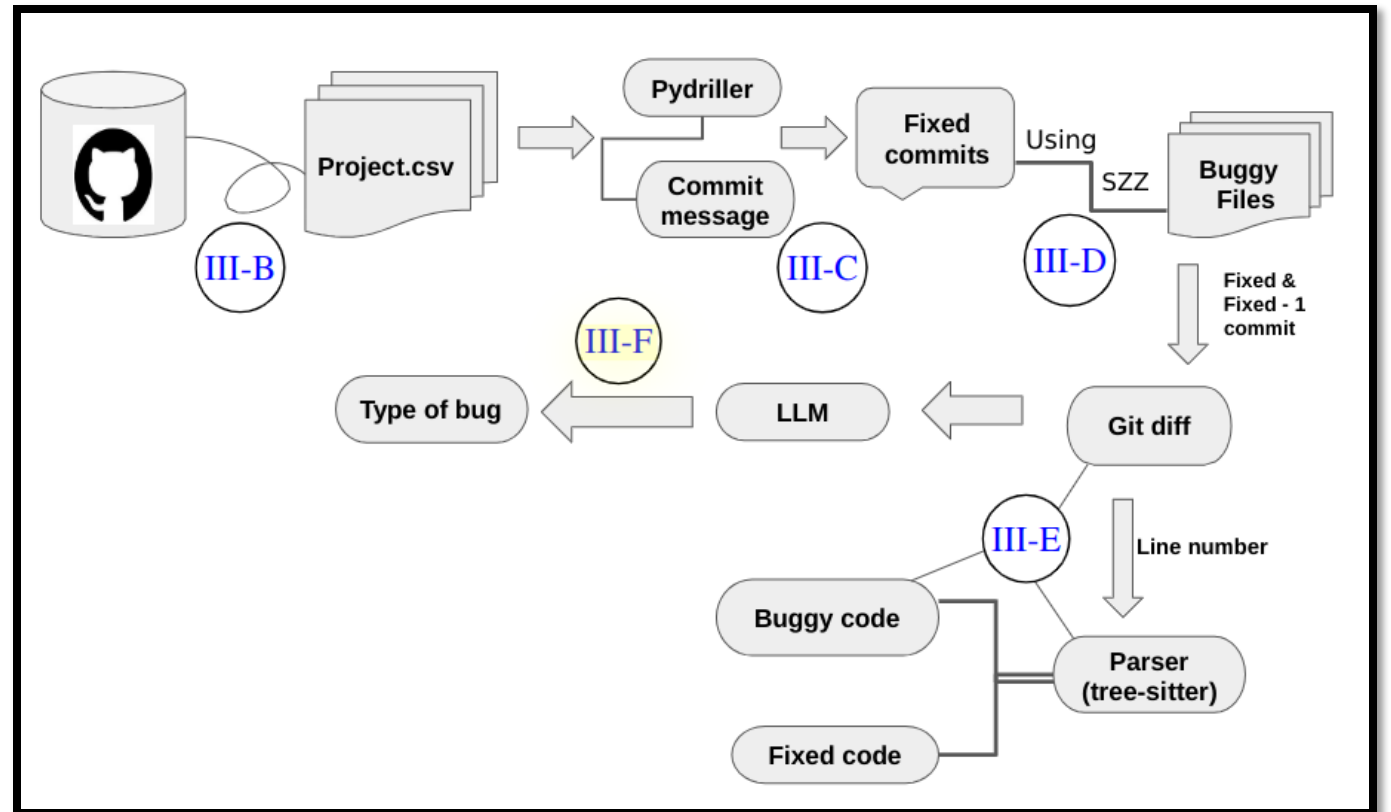
Terminology

- **GitHub** is a **web-based platform** commonly used for **version control** and **collaboration** in software development projects.
- A **commit** refers to a specific **revision** or **snapshot of a project's files** and directories. When a developer makes changes to the codebase, they create a commit to record those changes. A commit typically includes a **concise message** that **describes the purpose or nature of the changes made** - **Commit Message**

Bug Fix Dataset Acquisition Flowchart with Precise Context

Data acquisition is done in following ways:

- **Project Selection**
- Bug fix commits
- Extracting Bug and fixed code Snippets
- Type of the bug



Projects Selection Criteria (Visual Representation)

*In our project selection process, we implemented a rigorous and stringent set of filtering criteria. These criteria were designed to sift through a wide array of options and zero in on real-world projects. This approach ensured that the **projects we selected were not only substantial but also highly relevant to practical applications in the fields of C, C++, and Python.***

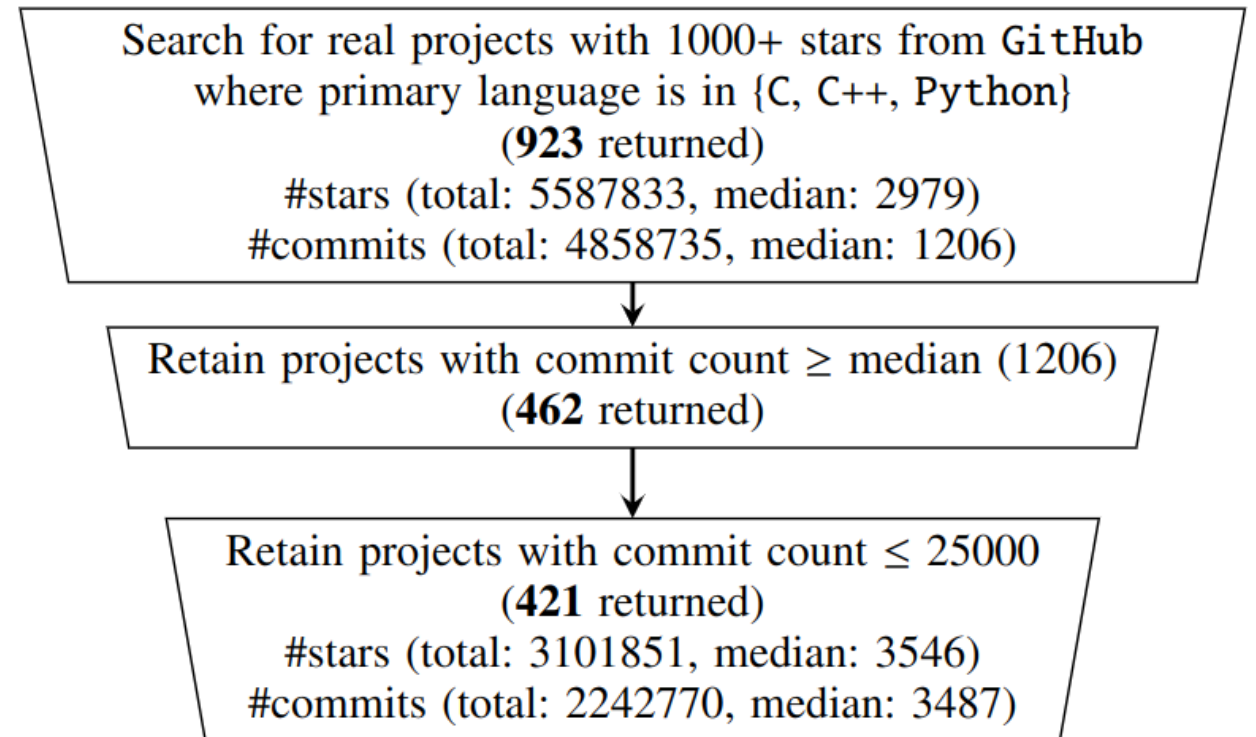


Fig. 2. Our project selection process.

Example workflow...

To demonstrate with examples, we chose a project named **Geocoder** which is written in **Python**

```
git diff
22 22 @property
23 23 def bbox(self):
24 24     extent = self.raw['properties'].get('extent')
25 -     if extent:
25 +     if extent and all(extent):
26 26         west = extent[0]
27 27         north = extent[1]
28 28         east = extent[2]
```



Geocoder: Simple, Consistent

Release v1.38.1. ([Installation](#))

Simple and consistent geocoding library written in Python.

Many online providers such as Google & Bing have geocoding services but do not include Python libraries and have different JSON responses between

It can be very difficult sometimes to parse a particular geocoding provider as one of them have their own JSON schema.

Here is a typical example of retrieving a Lat & Lng from Google using geocoder which shouldn't be this hard.

Geocoder is a simple and consistent geocoding library written in Python. Dealing with multiple different geocoding providers such as Google, Bing, OSM & many more has never been easier.

Support

If you are having issues we would love to hear from you. Just [hit me up](#). You can alternatively raise an issue [here](#) on Github.

```
>>> import requests
>>> url = 'https://maps.googleapis.com/maps/api/geocode/
>>> params = {'sensor': 'false', 'address': 'Mountain View
>>> r = requests.get(url, params=params)
>>> results = r.json()['results']
>>> location = results[0]['geometry']['location']
>>> location['lat'], location['lng']
(37.3860517, -122.0838511)
```

Now let's use Geocoder to do the same task.

```
>>> import geocoder
>>> g = geocoder.google('Mountain View, CA')
>>> g.latlng
(37.3860517, -122.0838511)
```

Before

```
18 @property
19 def lng(self):
20     return self.raw['geometry']['coordinates'][0]
21
22 @property
23 def bbox(self):
24     extent = self.raw['properties'].get('extent')
25 -     if extent:
26         west = extent[0]
27         north = extent[1]
28         east = extent[2]
29         south = extent[3]
30     return BBox.factory([south, west, north,
31                             east]).as_dict
32
33 @property
34 def address(self):
35     # Ontario, Canada
36     address = ', '.join([self.state, self.country])
```

After

```
18 @property
19 def lng(self):
20     return self.raw['geometry']['coordinates'][0]
21
22 @property
23 def bbox(self):
24     extent = self.raw['properties'].get('extent')
25 +     if extent and all(extent):
26         west = extent[0]
27         north = extent[1]
28         east = extent[2]
29         south = extent[3]
30     return BBox.factory([south, west, north,
31                             east]).as_dict
32
33 @property
34 def address(self):
35     # Ontario, Canada
36     address = ', '.join([self.state, self.country])
```

Loc: [before: 25, after: 25]

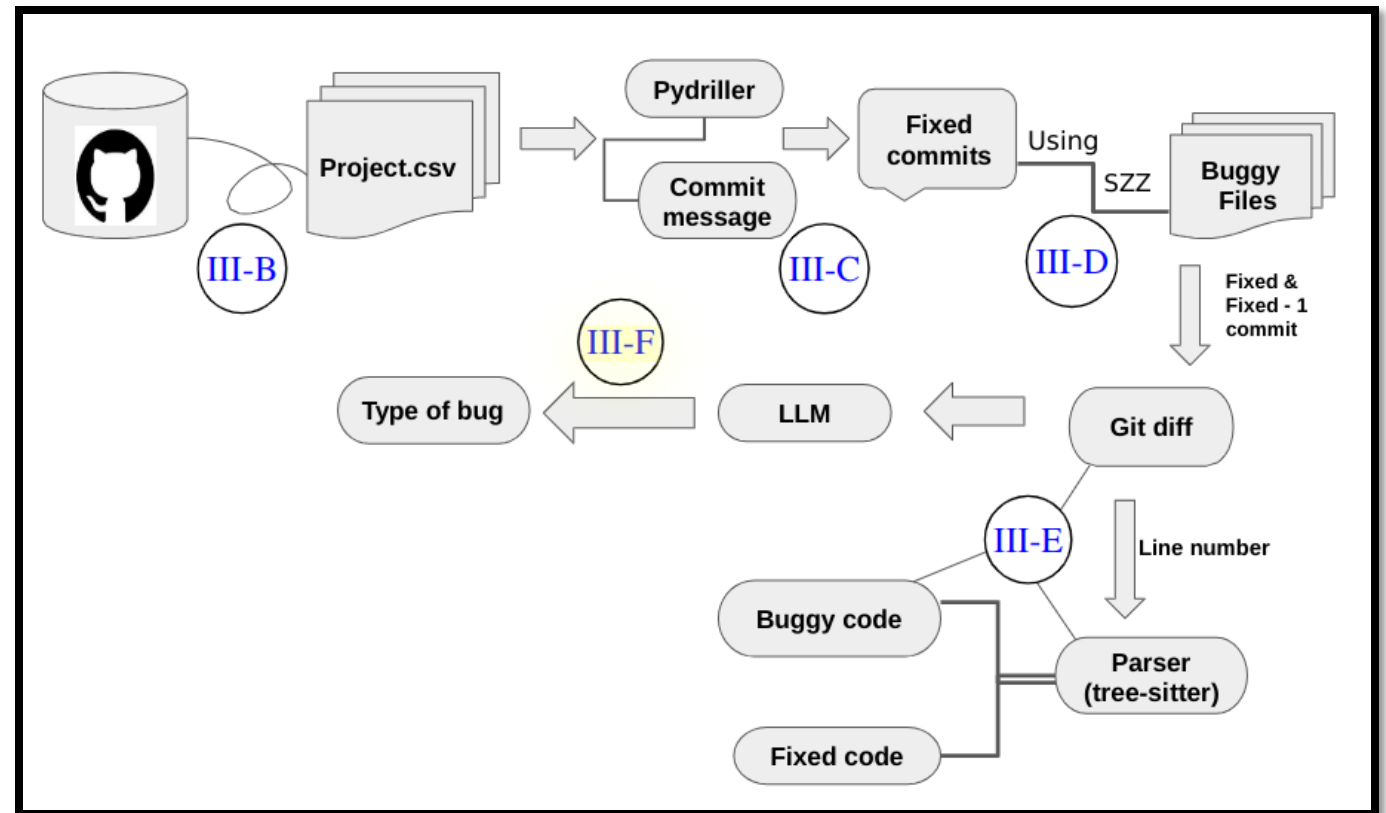
Bug Type: fix komoot.py when extent is none

Commit Message: fix calls to BBox with invalid input

Bug Fix Dataset Acquisition Flowchart with Precise Context

Data acquisition is done in following ways:

- Project Selection
- **Bug fix commits**
- Extracting Bug and fixed code Snippets
- Type of the bug



Collection of Bug-fix Commits

The collection of bug-fixing commits involved multiple strategies:

- Commit messages were analyzed to identify bug fix commits, leveraging keywords related to bug fixes. A total of **52 keywords** are used.
- Also, **Regular expressions** were utilized to perform precise searches for bug fix commit patterns
- The **SZZ algorithm** was applied to trace back and identify the buggy files associated with bug fix commits

“fixed ”, “ bug”, “fixes ”, “fix ”, “ fix”, “ fixed”, “ fixes”, “crash”, “solves”, “ resolves”, “resolves ”, “ issue”, “issue ”, “regression”, “fall back”, “assertion”, “coverity”, “reproducible”, “stack-wanted”, “steps-wanted”, “testcase”, “failur”, “fail”, “npe ”, “ npe”, “except”, “broken”, “differential testing”, “error”, “hang ”, “ hang”, “test fix”, “steps to reproduce”, “crash”, “assertion”, “failure”, “leak”, “stack trace”, “heap overflow”, “freez”, “problem ”, “ problem”, “ overflow”, “overflow ”, “avoid ”, “ avoid”, “workaround ”, “ workaround”, “break ”, “ break”, “ stop”, “stop ”

Commit messages with
52 keywords [6] (c.f. Sec. III-C)

```
+ RegEx. [7]: ‘.((solv(ed|es|e|ing))  
|(fix(s|es|ing|ed)?))’  
|((error|bug|issue)(s)?)).’
```

→ SZZ Algorithm [8]

[J. Sliwinski, T. Zimmermann, and A. Zeller; (2005)] SZZ takes a commit (**bug-fixing**) as the input and returns a list of commits (**bug-introducing**) that last **added** the **deleted** lines in the input commit.

Example workflow...

To demonstrate with examples, we chose a project named **Geocoder** which is written in **Python**

```
geocoder/komoot.py  git diff
-22,7 +22,7 @@ def lng(self):
22 22     @property
23 23     def bbox(self):
24 24         extent = self.raw['properties'].get('extent')
25 -         if extent:
26 +         if extent and all(extent):
27         west = extent[0]
28         north = extent[1]
29         east = extent[2]
```



Geocoder: Simple, Consistent

Release v1.38.1. (Installation)

Simple and consistent geocoding library written in Python.

Many online providers such as Google & Bing have geocoding services, these providers do not include Python libraries and have different JSON responses between each other.

It can be very difficult sometimes to parse a particular geocoding provider since each one of them have their own JSON schema.

Here is a typical example of retrieving a Lat & Lng from Google using Python, things shouldn't be this hard.

```
>>> import requests
>>> url = 'https://maps.googleapis.com/maps/api/geocode/json'
>>> params = {'sensor': 'false', 'address': 'Mountain View, CA'}
>>> r = requests.get(url, params=params)
>>> results = r.json()['results']
>>> location = results[0]['geometry']['location']
>>> location['lat'], location['lng']
(37.3860517, -122.0838511)
```

Support

If you are having issues we would love to hear from you. Just hit me up. You can alternatively raise an issue here on Github.

Now lets use Geocoder to do the same task.

```
>>> import geocoder
>>> g = geocoder.google('Mountain View, CA')
>>> g.latlng
(37.3860517, -122.0838511)
```

```
18 @property
19 def lng(self):
20     return self.raw['geometry']['coordinates'][0]
21
22 @property
23 def bbox(self):
24     extent = self.raw['properties'].get('extent')
25 -     if extent:
26         west = extent[0]
27         north = extent[1]
28         east = extent[2]
29         south = extent[3]
30         return BBox.factory([south, west, north,
31                               east]).as_dict
32
33 @property
34 def address(self):
35     # Ontario, Canada
36     address = ', '.join([self.state, self.country])
```

Before

```
18 @property
19 def lng(self):
20     return self.raw['geometry']['coordinates'][0]
21
22 @property
23 def bbox(self):
24     extent = self.raw['properties'].get('extent')
25 +     if extent and all(extent):
26         west = extent[0]
27         north = extent[1]
28         east = extent[2]
29         south = extent[3]
30         return BBox.factory([south, west, north,
31                               east]).as_dict
32
33 @property
34 def address(self):
35     # Ontario, Canada
36     address = ', '.join([self.state, self.country])
```

After

Loc: [before: 25, after: 25]

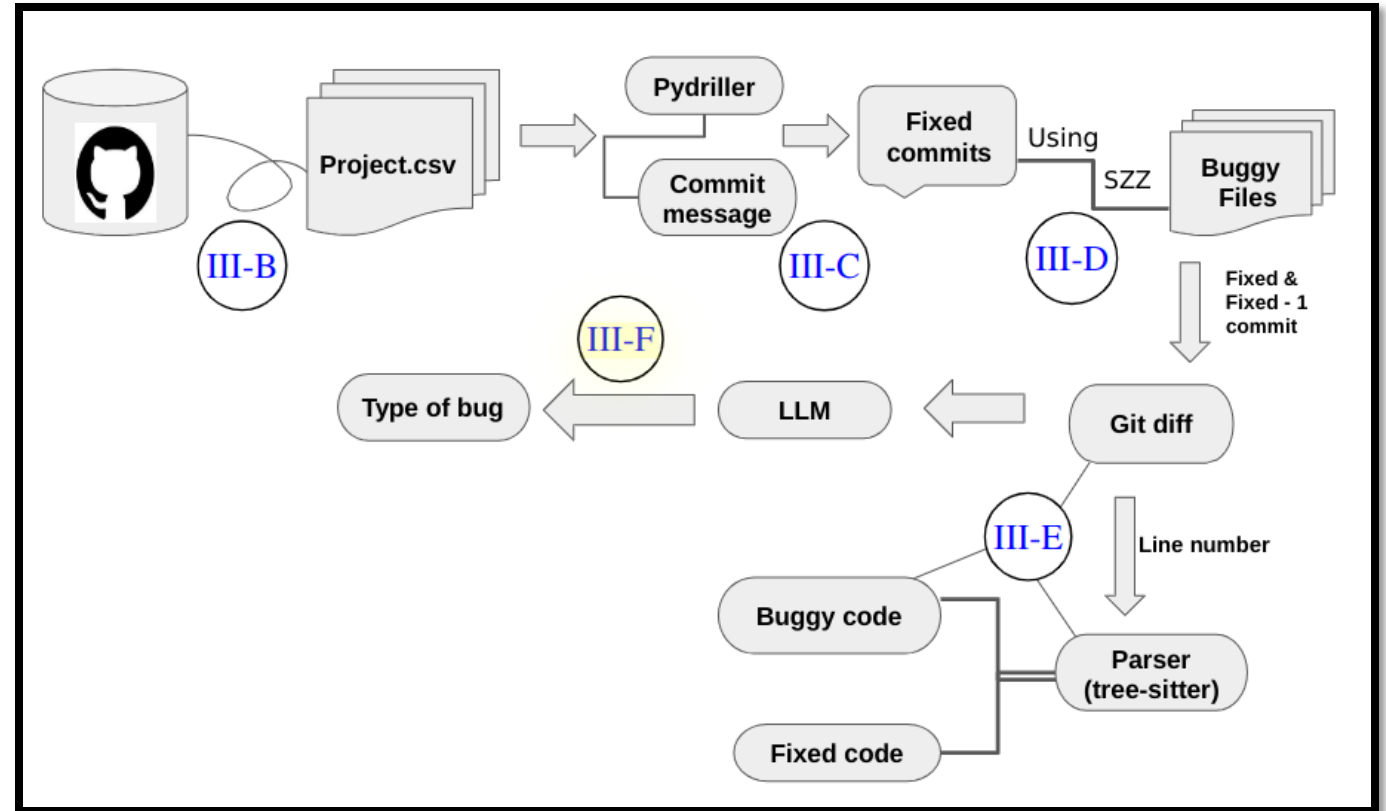
Bug Type: fix komoot.py when extent is none

Commit Message: fix calls to BBox with invalid input

Bug Fix Dataset Acquisition Flowchart with Precise Context

Data acquisition is done in following ways:

- Project Selection
- Bug fix commits
- **Extracting Bug and fixed code Snippets**
- Type of the bug



Extracting Code Snippets

The extraction of code snippets is done as follows:

Step 1:

- **Location:** The location of bug is found using `git diff` between `fixed - 1` and `fixed` commit. The location here indicates the **line number** in which the bug is detected and fixed.

a **modification**
typically involves an
addition after a **deletion**

Loc: [before: 25, after: 25]

Bug Type: fix komoot.py when extent is none

Commit Message: fix calls to BBox with invalid input

Extracting Code Snippets

Step 2:

- **Buggy Code:** The location[before] obtained is used to extract the code snippet using **Tree-Sitter** parser.
- **Fixed Code:** The location[after] obtained is used to extract the code snippet using **Tree-Sitter** parser.

```
18     @property
19     def lng(self):
20         return self.raw['geometry']['coordinates'][0]
21
22     @property
23     def bbox(self):
24         extent = self.raw['properties'].get('extent')
25 -     if extent:
26         west = extent[0]
27         north = extent[1]
28         east = extent[2]
29         south = extent[3]
30         return BBox.factory([south, west, north,
31                             east]).as_dict
32
33     @property
34     def address(self):
35         # Ontario, Canada
36         address = ', '.join(
```

```
18     @property
19     def lng(self):
20         return self.raw['geometry']['coordinates'][0]
21
22     @property
23     def bbox(self):
24         extent = self.raw['properties'].get('extent')
25 +     if extent and all(extent):
26         west = extent[0]
27         north = extent[1]
28         east = extent[2]
29         south = extent[3]
30         return BBox.factory([south, west, north,
31                             east]).as_dict
32
33     @property
34     def address(self):
35         if state, self.country]]
```

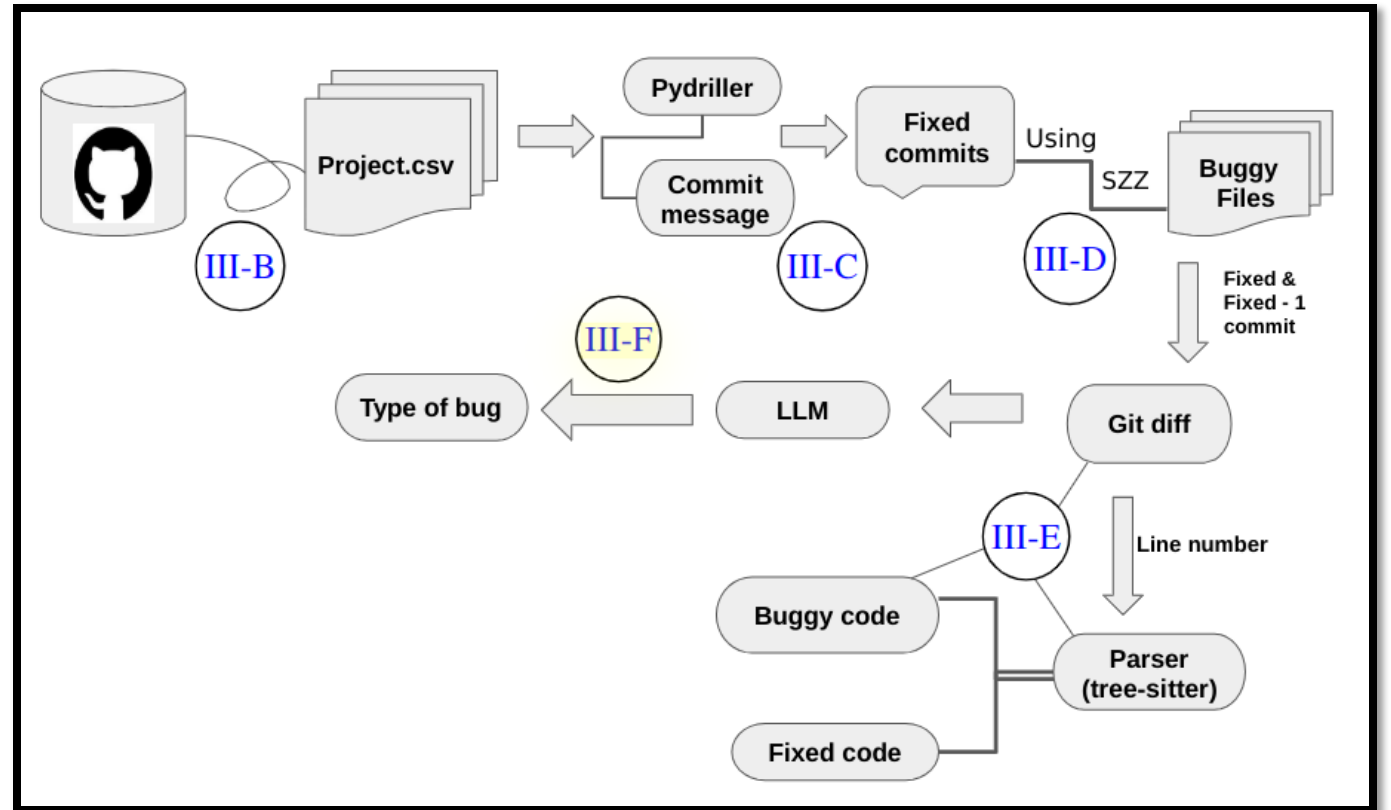
Loc: [before: 25, after: 25]

Bug Type: fix komoot.py when extent is none
Commit Message: fix calls to BBox with
invalid input

Bug Fix Dataset Acquisition Flowchart with Precise Context

Data acquisition is done in following ways:

- Project Selection
- Bug fix commits
- Extracting Bug and fixed code Snippets
- **Type of the bug**



Inferencing the type of Bug

The type of bug is inferenced using a **commit message generator** Large Language Model (LLM).

URL: <https://huggingface.co/mamiksik/CommitPredictorT5>

The LLM takes `git-diff` of fixed - 1 and fixed commits as input and generates the summary of diff which is nothing but a fix and thus inferred as **bug type**.

Hugging Face Search models, datasets, users...

mamiksik **CommitPredictorT5** like 0

Text2Text Generation Transformers PyTorch t5 generated_from_trainer AutoTrain Compatible text-generation-inference License: bsd-3-clause

Model card Files and versions Community

Edit model card

Downloads last month
0

Hosted inference API ⓘ
Text2Text Generation

Your sentence here...

Loc: [before: 25, after: 25]
Bug Type: fix komoot.py when extent is none
Commit Message: fix calls to BBox with invalid input

JSON Output Maximize

CommitPredictorT5

This model is a fine-tuned version of [Salesforce/codet5-base-multi-sum](#) on the None dataset. It achieves the following results on the evaluation set:

- Loss: 2.4669
- Bleu: 0.0002
- Precisions: [0.003189792663476874, 0.00016826518593303046, 0.000321853878339234, 0.0036900369003690036]
- Brevity Penalty: 0.2394
- Length Ratio: 0.4116
- Translation Length: 10658
- Reference Length: 25896

Texts, References, and Acknowledgements

Online:

- Continuous Integration and Delivery (**CircleCI**: <https://circleci.com>)

Textbook:

- Sharp, J. (2022). *Microsoft Visual C# Step by Step*, 10th edition, Microsoft Press.
- Watson, K., Nagel, C., Pedersen, J. H., Reid, J. D., & Skinner, M. (2008). *Beginning Microsoft Visual C# 2008*. John Wiley & Sons.
- Mark J. Price (2024). *C# 13 and .NET 9 – Modern Cross-Platform Development Fundamentals*, 9th edition, Packt Publishing Ltd.

Reference:

- Soni, M. (2016). *DevOps for Web Development*. Packt Publishing Ltd.
- Yusuf Sulisty Nugroho, Hideaki Hata, and Kenichi Matsumoto. 2020. *How different are different diff algorithms in Git? Use --histogram for code changes*. Empirical Softw. Engg. 25, 1 (Jan 2020), 790–823.
- Sai Krishna Avula (2024), *Mining Software Bug Fixes and SAT Analysis: Dataset Creation and Tool Development for Improved Software Quality Assurance* - Awarded Gold Medal for the outstanding research (M.Tech.) at 13th Convocation IITGN.