CS202: Software Tools and Techniques for CSE

Lecture 3

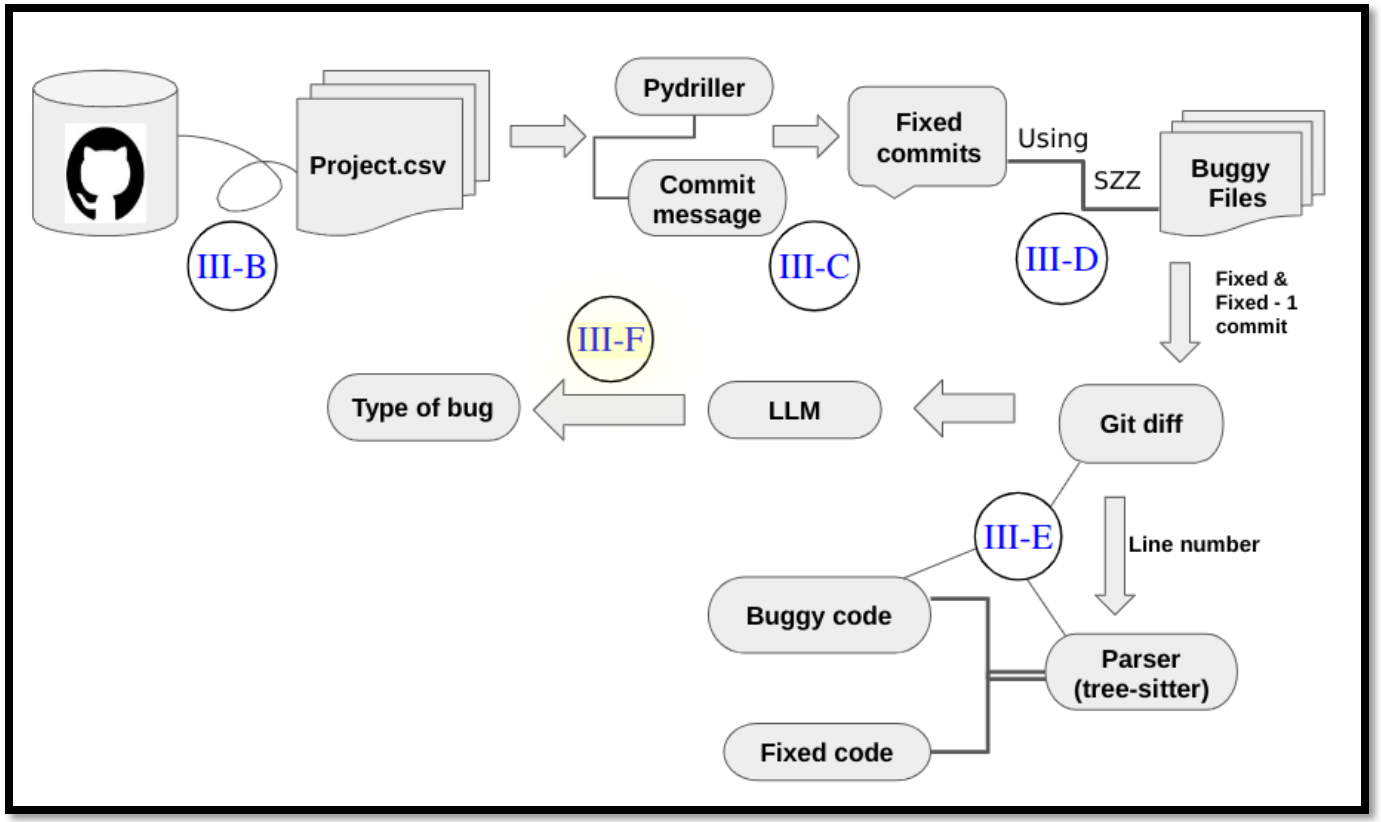
Shouvick Mondal

shouvick.mondal@iitgn.ac.in
August 2025

Bug Fix Dataset Acquisition Flowchart with Precise Context

Data acquisition is done in following ways:

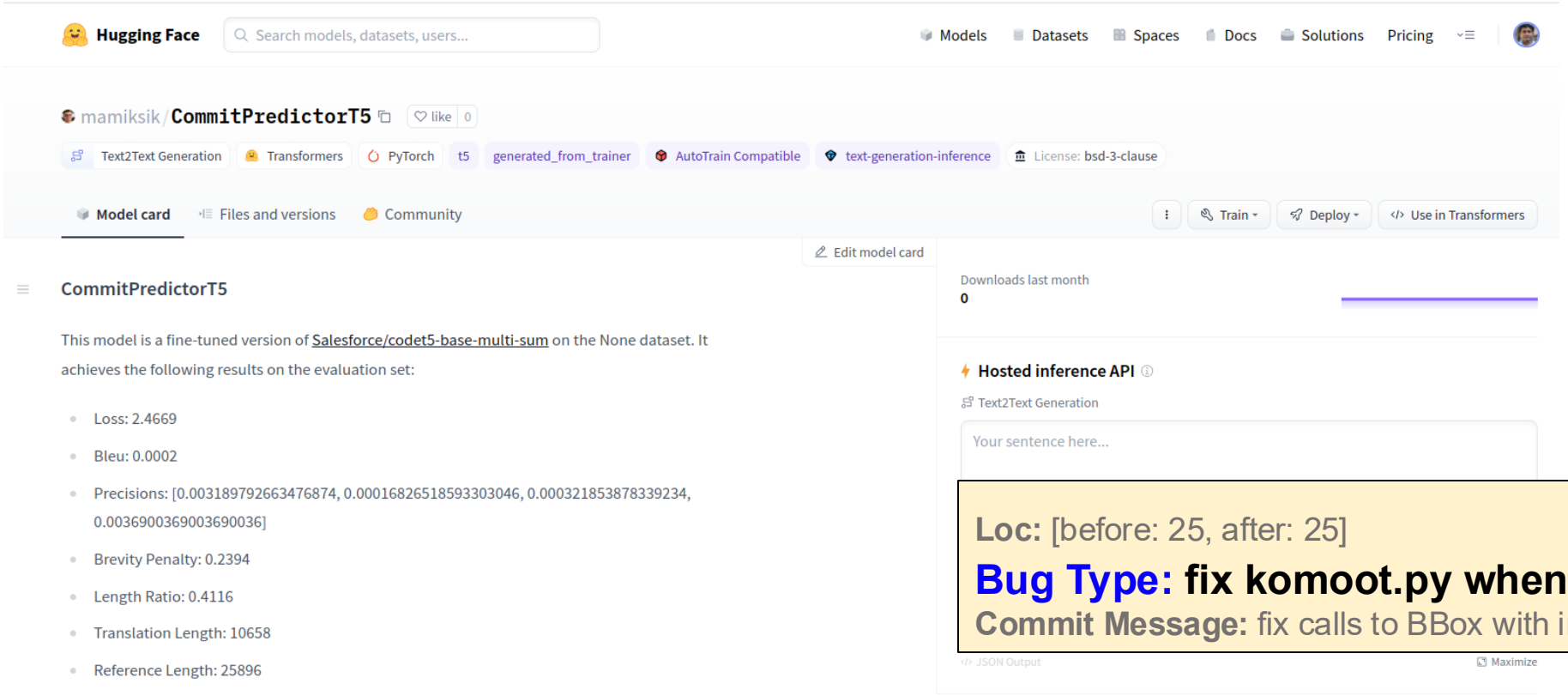
- Project Selection
- Bug fix commits
- Extracting Bug and fixed code Snippets
- **Type of the bug**



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.



Loc: [before: 25, after: 25]

Bug Type: fix komoot.py when extent is none

Commit Message: fix calls to BBox with invalid input

JAVA Example

Before Bug fix:

```
31  * Functional tests for the {@link
        AutoFactoryProcessor}.
32  */
33  public class AutoFactoryProcessorTest {
34  @Test public void simpleClass() {
35  ASSERT.about(javaSource())
36 .that(...("tests/SimpleClass.java"))
37  .processedWith(new AutoFactoryProcessor())
38  .compilesWithoutError()
39  .and().generatesSources(JavaFileObjects
40  ...("tests/SimpleClassFactory.java"));
41  }
42
43  @Test public void publicClass() {
```

After Bug fix:

```
30  * Functional tests for the {@link
        AutoFactoryProcessor}.
31  */
32  public class AutoFactoryProcessorTest {
33  @Test public void simpleClass() {
34  ASSERT.about(javaSource())
35  .that(...("good/SimpleClass.java"))
36  .processedWith(new AutoFactoryProcessor())
37  .compilesWithoutError()
38  .and().generatesSources(JavaFileObjects...
39  ("expected/SimpleClassFactory.java"));
40  }
41
42  @Test public void publicClass() {
```

Loc: [before: 36, after: 35]
Bug Type: fix coverage for autofactoryprocessor tests
Commit Message: Break up the test files so that it's clear which test files are expected to work.

C++ Example

Before Bug Fix:

```
287 }
288
289
290 static void gc_allocation_map_remove(...)
291 {
292     // ignores unknown keys
293     size_t index = gc_hash(ptr) % am->capacity;
294     Allocation* cur = am->allocs[index];
295     Allocation* prev = NULL;
296     while(cur != NULL) {
297         if (cur->ptr == ptr) {
298             // found it
299             if (!prev) {
300                 // first item in list
301                 am->allocs[index] = cur->next;
302             } else {
303                 // not the first item in the list
304                 prev->next = cur->next;
305             }
306             gc_allocation_delete(cur);
307             am->size--;
308         } else {
309             // move on
310             prev = cur;
311         }
312         cur = cur->next;
313     }
314     if (allow_resize) {
315         gc_allocation_m
316     }
317 }
```

After Bug Fix:

```
287 }
288
289 }
290
291
292 static void gc_allocation_map_remove(...)
293 {
294     // ignores unknown keys
295     size_t index = gc_hash(ptr)  am->capacity;
296     Allocation* cur = am->allocs[index];
297     Allocation* prev = NULL;
298     while(cur != NULL) {
299         Allocation* cur_next = cur->next;
300         if (cur->ptr == ptr) {
301             // found it
302             if (!prev) {
303                 // first item in list
304                 am->allocs[index] = cur->next;
305             } else {
306         // not the first item in the list
307                 prev->next = cur->next;
308             }
309             gc_allocation_delete(cur);
310             am->size--;
311         } else {
312             // move on
313             prev = cur;
314         }
315         cur = cur_next;
316     }
317     if (allow_resize) {
```

Loc: [before: 314, after: 315]

Bug Type: use cur_next instead of cur->next in gc_allocation_map_remove

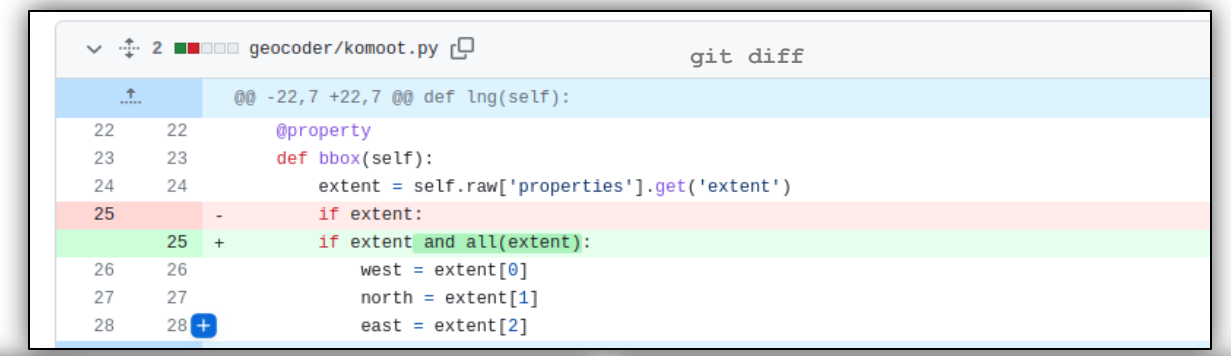
Commit Message: Crash fixing.

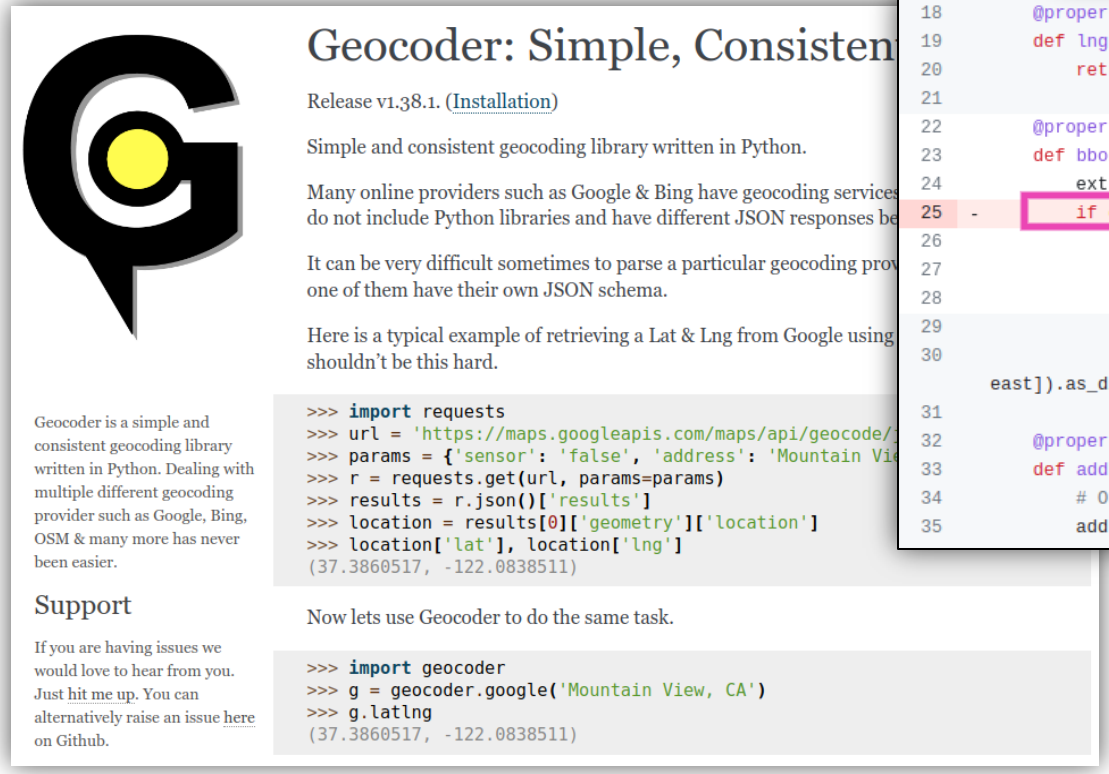
Context of a bug

- specific **circumstances** or **conditions** in which a bug **occurs** or **manifests** itself.
- Conditions are typically represented by enclosing code constructs in the source language
- sequence of statements leading to the bug, and the dependencies between different code modules or functions

Within function

Here the context of bug is the function in which the bug is present and three lines below and above the function

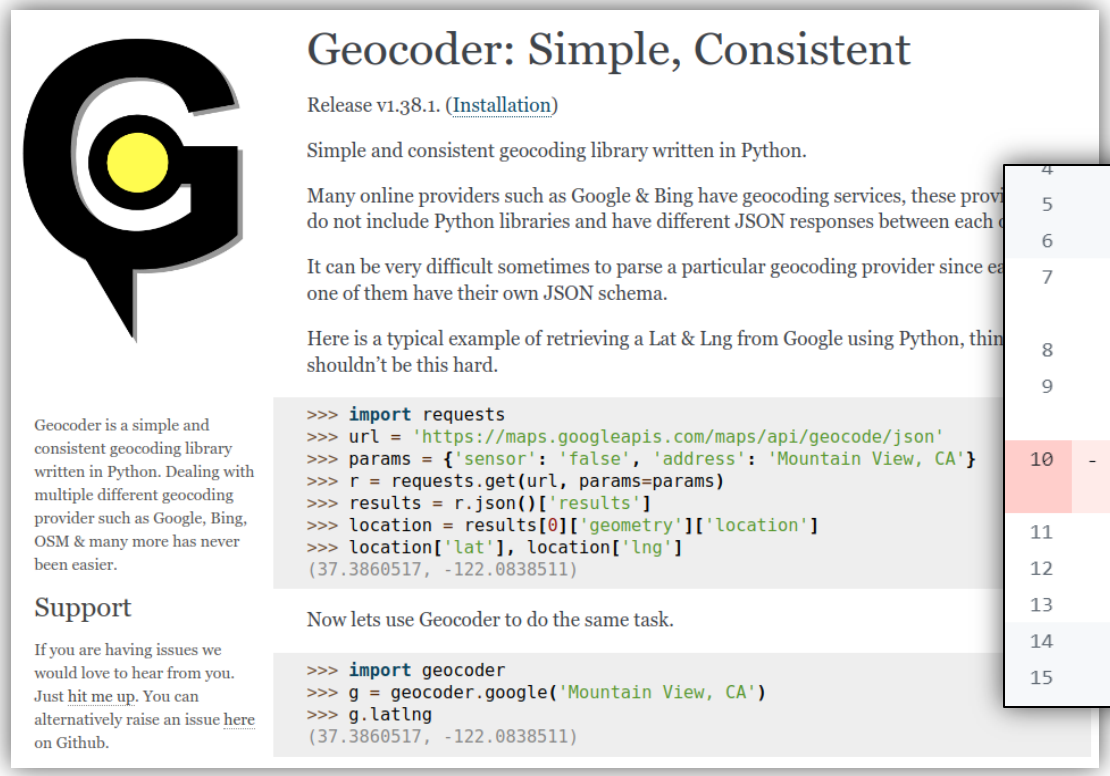




Loc: [before: 25, after: 25]
Bug Type: fix komoot.py when extent is none
Commit Message: fix calls to BBox with invalid input

Outside function

The **context** of the bug here is five lines above and below it.



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 provi
do not include Python libraries and have different JSON responses between each

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

Here is a typical example of retrieving a Lat & Lng from Google using Python, thin
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)
```

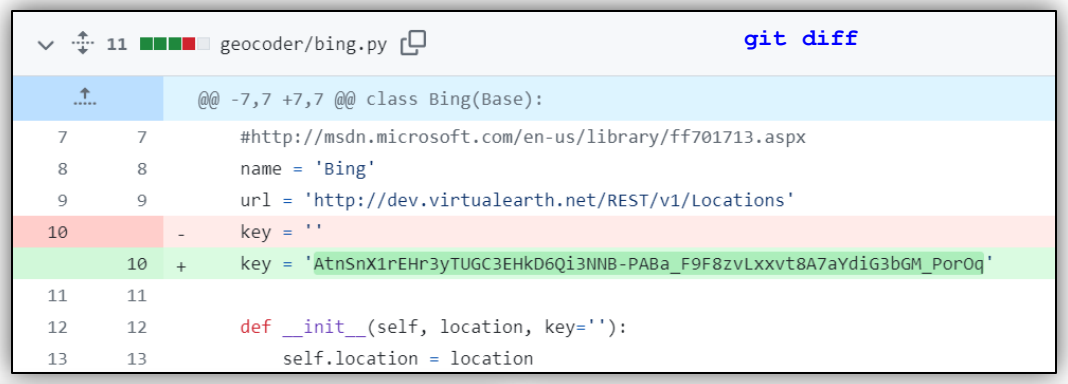
Now lets use Geocoder to do the same task.

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

Geocoder is a simple and consistent geocoding library written in Python. Dealing with multiple different geocoding provider 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.



11 geocoder/bing.py — git diff

```
@@ -7,7 +7,7 @@ class Bing(Base):
7  7      #http://msdn.microsoft.com/en-us/library/ff701713.aspx
8  8      name = 'Bing'
9  9      url = 'http://dev.virtualearth.net/REST/v1/Locations'
10    -    key = ''
   10   +  key = 'AtnSnX1rEHr3yTUGC3EHkD6Qi3NNB-PABa_F9F8zvLxxvt8A7aYdiG3bGM_PorOq'
11 11
12 12      def __init__(self, location, key=''):
13 13          self.location = location
```

Before

```
5
6   class Bing(Base):
7       #http://msdn.microsoft.com/en-
    us/library/ff701713.aspx
8       name = 'Bing'
9       url =
    'http://dev.virtualearth.net/REST/v1/Locations'
10  -    key = ''
11
12      def __init__(self, location, key=''):
13          self.location = location
14          if not key:
15              key = self.key
```

After

```
5
6   class Bing(Base):
7       #http://msdn.microsoft.com/en-
    us/library/ff701713.aspx
8       name = 'Bing'
9       url =
    'http://dev.virtualearth.net/REST/v1/Locations'
10  +    key = 'AtnSnX1rEHr3yTUGC3EHkD6Qi3NNB-
    PABa_F9F8zvLxxvt8A7aYdiG3bGM_PorOq'
11
12      def __init__(self, location, key=''):
13          self.location = location
14          if not key:
15              key = self.key
```

Loc: [before: 10, after: 10]
Bug Type: add key to bing
Commit Message: Fixed bbox & Added keys

What is Precise Code Context ?

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,
    east]).as_dict
31
32      @property
33      def address(self):
34          # Ontario, Canada
35          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,
    east]).as_dict
31
32      @property
33      def address(self):
34          # Ontario, Canada
35          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

Combining everything about bug makes it **precise code context**

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

- Automatic **language-agnostic** algorithm (**C/C++**, **Java** and **Python**)
- Extract code snippets before and after bug fix
- **Precise location** of bugs (file, function and statement (line no.)
- Bug type
- A **dataset** of bug-fix pairs

COMPARISON OF MINECRAFT WITH EXISTING REPOSITORY MINED DATASETS

	FixJs [3]	*Tufano et al.* [4]	*Minecraft* (this paper)
Introduced in	MSR 2022	TOSEM 2019	This paper
Language	JavaScript	Java	C, C++, Python, and Java
Granularity	Function	Function	File, Function, and Statement (ranges)
#Commits	~2M	~787K	~2.2M
#Bug-Fixes	300K	~2.3M	~3.29M
Dataset size	5.47GB	~7GB	28.8GB
Bug Detection Strategy	Commit messages with 6 keywords [5]: ["fix", "solve", "bug", "issue", "problem", "error"]	Commit messages with 6 keywords [5]: ["fix", "solve", "bug" "issue", "problem", "error"]	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]

Dataset

- Has **five** columns
- Size of dataset: **28.8GB**
- A total of ~3.29M bug-fix pairs (C/C++, Python, Java)
  - 421 projects

Col#	Name	Type	Description
1	`Before Bug fix`	String	Code lines (+ context) before bug fix
2	`After Bug fix`	String	Code lines (+ context) after bug fix
3	`Location`	List of ints.	Line #s of the bug and it's fix
4	`Bug type`	String	Bug type inferred from LLM using `git dif`
5	`Commit Message`	String	Code review and version control

https://zenodo.org/records/8164641

Minecraft

"Concrete workflow. The proposed workflow can automatically mine the bug fixes from GitHub repositories, with specific information that basically meets the expectations"

-Reviewer (ASE)

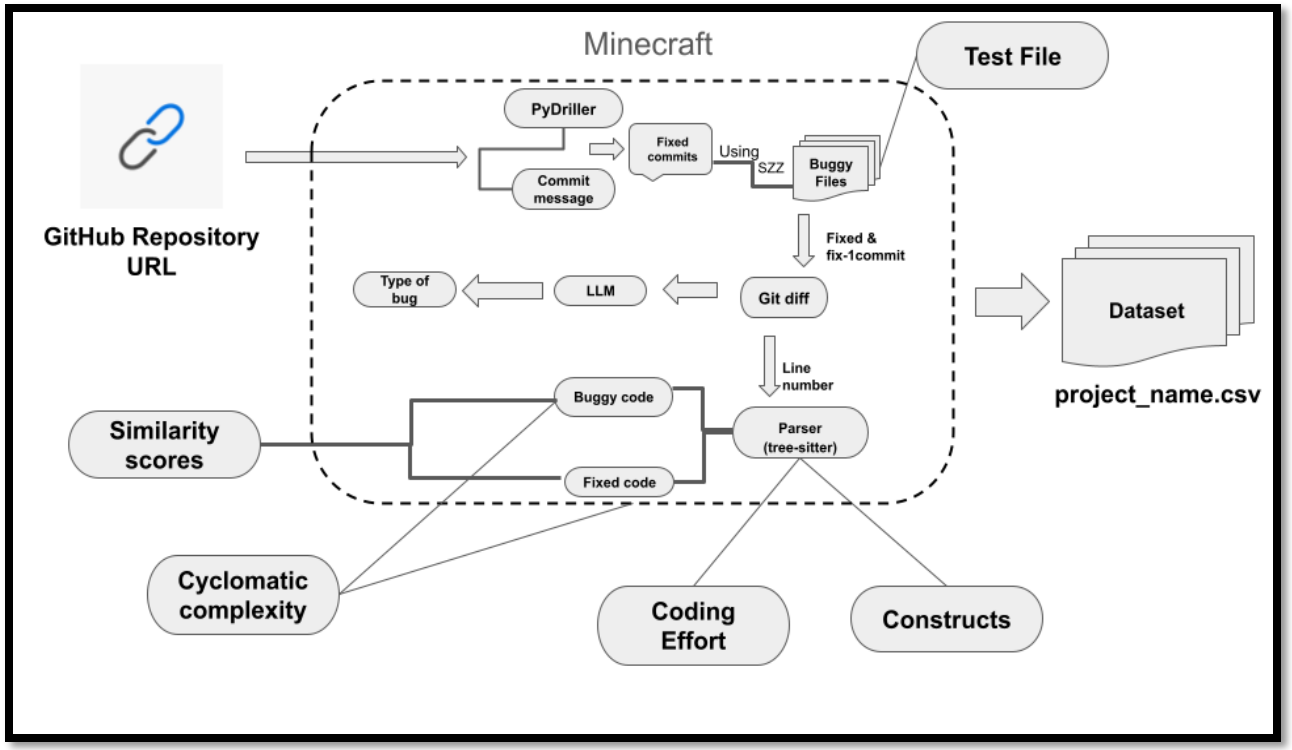
MineCPP: Mining Bug Fix Pairs and Their Structures

Published in FSE 2024 (CORE A*) Demonstrations track

https://2024.esec-fse.org/track/fse-2024-demonstrations

MineCPP: An extension of Minecraft

- Minecraft overcame limitations of existing datasets
- Is the data produced useful enough for comprehensive bug analysis?
- **MineCPP (Minecraft ++)** - extend the capabilities of bug mining methodology (Minecraft) by introducing **innovative features** and **metrics** not present in existing tools.



MineCPP: Additional Features

- **Coding Effort**
- **Test Case indicator**
- **Constructs surrounding the bug**
- **Cyclomatic Complexity**
- **Similarity Score**

MineCPP: Additional Features

- **Coding Effort**
- **Test Case indicator**
- **Constructs surrounding the bug**
- **Cyclomatic Complexity**
- **Similarity Score**

Coding Effort

- **Novel metric** introduced in MineCPP
- The **#nodes** traversed before reaching the **buggy node** of an Abstract Syntax Tree (**AST**)
- Quantifies the complexity and effort required by author/developer to introduce bugs within source code.

Example: Coding Effort

- Buggy code for **addition**
- **AST** of Buggy Code
- Buggy node
- AST nodes Traversal till buggy node
- Coding Effort: **24**

```
1  import sys
2
3  def add(a: int, b: int):
4      return a - b
5
6  num1 = int(input())
7  num2 = int(input())
8  print(add(num1, num2))
```

Buggy Code

```
module (0, 4)
  import_statement (0, 0)
    import (0, 0)
    dotted_name (0, 0)
      identifier (0, 0)
  function_definition (2, 3)
    def (2, 2)
    identifier (2, 2)
    parameters (2, 2)
      ( (2, 2)
      typed_parameter (2, 2)
        identifier (2, 2)
        : (2, 2)
        type (2, 2)
          identifier (2, 2)
      , (2, 2)
      typed_parameter (2, 2)
        identifier (2, 2)
        : (2, 2)
        type (2, 2)
          identifier (2, 2)
      ) (2, 2)
    : (2, 2)
    block (3, 3)
      return_statement (3, 3)
        return (3, 3)
        binary_operator (3, 3)
          identifier (3, 3)
          - (3, 3)
          identifier (3, 3)
```

AST

Module -> import_statement -> import -> dotted_name -> identifier-> function_definition -> def -> identifier -> parameters -> (-> typed_parameter -> identifier -> : -> type -> identifier -> , -> typed_parameter -> identifier -> : -> type-> identifier ->) -> : -> block

MineCPP: Additional Features

- Coding Effort
- **Test Case indicator**
- Constructs surrounding the bug
- Cyclomatic Complexity
- Similarity Score

Test Case Indicator

- Indicates whether a bug-fix pair has a test case. **1** indicates presence and **0** indicates absence
- Navigates through the repository files to match regular expressions against files to find a test case.

```
re.compile(r'test[^a-zA-Z]*{}|{}[^a-zA-Z]*test'

.format(re.escape(base_fileName_withoutExtension),

re.escape(base_fileName_withoutExtension)), re.IGNORECASE)
```

Regular Expression

MineCPP: Additional Features

- Coding Effort
- Test Case indicator
- **Constructs surrounding the bug**
- Cyclomatic Complexity
- Similarity Score

Constructs surrounding the bug

- The “**Constructs**” metric **focuses** language constructs surrounding the bug context
- The metric considers the **frequency** of these constructs within the immediate **context** of the bug.
- Provides targeted insights into the nature and characteristics of bugs within software codebases.

Example: Constructs

- Buggy code
- **Context** of Buggy Code
- AST of bug and its context
- **Constructs**

```
1  import sys
2
3  def add(a: int, b: int):
4      return a - b
5
6  num1 = int(input())
7  num2 = int(input())
```

Buggy Code (context)

```
module (0, 7)
  import_statement (0, 0)
    import (0, 0)
    dotted_name (0, 0)
      identifier (0, 0)
  function_definition (2, 3)
    def (2, 2)
    identifier (2, 2)
    parameters (2, 2)
      ( (2, 2)
      typed_parameter (2, 2)
        identifier (2, 2)
        : (2, 2)
        type (2, 2)
          identifier (2, 2)
      , (2, 2)
      typed_parameter (2, 2)
        identifier (2, 2)
        : (2, 2)
        type (2, 2)
          identifier (2, 2)
      ) (2, 2)
    : (2, 2)
    block (3, 3)
      return_statement (3, 3)
        return (3, 3)
        binary_operator (3, 3)
          identifier (3, 3)
          - (3, 3)
          identifier (3, 3)
  expression_statement (5, 5)
    assignment (5, 5)
      identifier (5, 5)
      = (5, 5)
      call (5, 5)
        identifier (5, 5)
        argument_list (5, 5)
          ( (5, 5)
          call (5, 5)
            identifier (5, 5)
            argument_list (5, 5)
              ( (5, 5)
              ) (5, 5)
          ) (5, 5)
  expression_statement (6, 6)
    assignment (6, 6)
      identifier (6, 6)
      = (6, 6)
      call (6, 6)
        identifier (6, 6)
        argument_list (6, 6)
          ( (6, 6)
          call (6, 6)
            identifier (6, 6)
            argument_list (6, 6)
              ( (6, 6)
              ) (6, 6)
          ) (6, 6)
```

AST

{**'module'**: 1, **'import_statement'**: 1, **'import'**: 1, **'dotted_name'**: 1, **'identifier'**: 14, **'function_definition'**: 1, **'def'**: 1, **'parameters'**: 1, **'('**: 5, **'typed_parameter'**: 2, **':'**: 3, **'type'**: 2, **','**: 1, **')'**: 5, **'block'**: 1, **'return_statement'**: 1, **'return'**: 1, **'binary_operator'**: 1, **'-'**: 1, **'expression_statement'**: 1, **'assignment'**: 2, **'='**: 2, **'call'**: 4, **'argument_list'**: 4}

MineCPP: Additional Features

- Coding Effort
- Test Case indicator
- Constructs surrounding the bug
- **Cyclomatic Complexity**
- Similarity Score

Cyclomatic Complexity

- The structural complexity of the code - the **#independent paths** through the source code
- Determines the **stability** and **maintainability** of the codebase following the resolution of the identified issue
- **Lizard** library
- Cyclomatic Complexity

  (**before** and **after**): **1**

```
1  import sys
2
3  def add(a: int, b: int):
4      return a - b
5
6  num1 = int(input())
7  num2 = int(input())
8  print(add(num1, num2))
```

Buggy Code

```
1  import sys
2
3  def add(a: int, b: int):
4      return a + b
5
6  num1 = int(input())
7  num2 = int(input())
8  print(add(num1, num2))
```

Bug-fix Code

MineCPP: Additional Features

- Coding Effort
- Test Case indicator
- Constructs surrounding the bug
- Cyclomatic Complexity
- **Similarity Score**

Similarity Score

- Provides a percentage of **code similarity** between the buggy and fixed code snippets
- **BLEU**, **CrystalBLEU**, and **Code BERT Score** used to calculate code similarity

- BLEU – **0.924**
- CrystalBLEU – **0.935**
- Code BERT Score – **0.99**

```
1  import sys
2
3  def add(a: int, b: int):
4      return a - b
5
6  num1 = int(input())
7  num2 = int(input())
8  print(add(num1, num2))
```

Buggy Code

```
1  import sys
2
3  def add(a: int, b: int):
4      return a + b
5
6  num1 = int(input())
7  num2 = int(input())
8  print(add(num1, num2))
```

Bug-fix Code

MineCPP (tool): Installation and Usage

- MineCPP is a **Python** installable package and can be installed using

pip install minecpp

Usage:

Optional arguments: `minecpp [-h] [-version] [-u U]`

Argument	Description
-h, -help	Shows tool's description and usage
-version	Shows tool's version
-u U	Provide the GitHub repo link to analyze

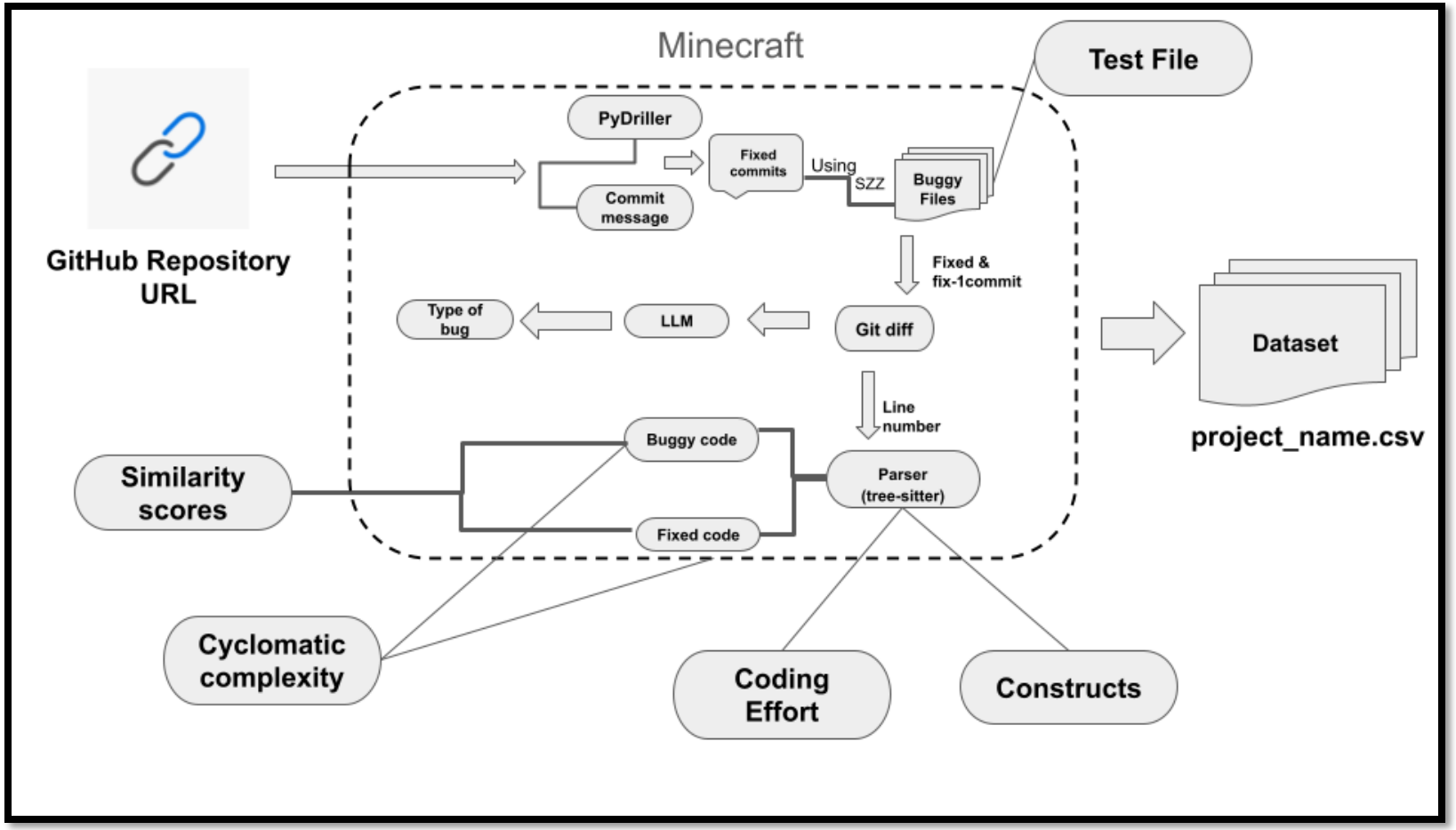
minecpp –u https://github.com/DenisCarriere/geocoder

https://pypi.org/project/Minecpp

Running the tool

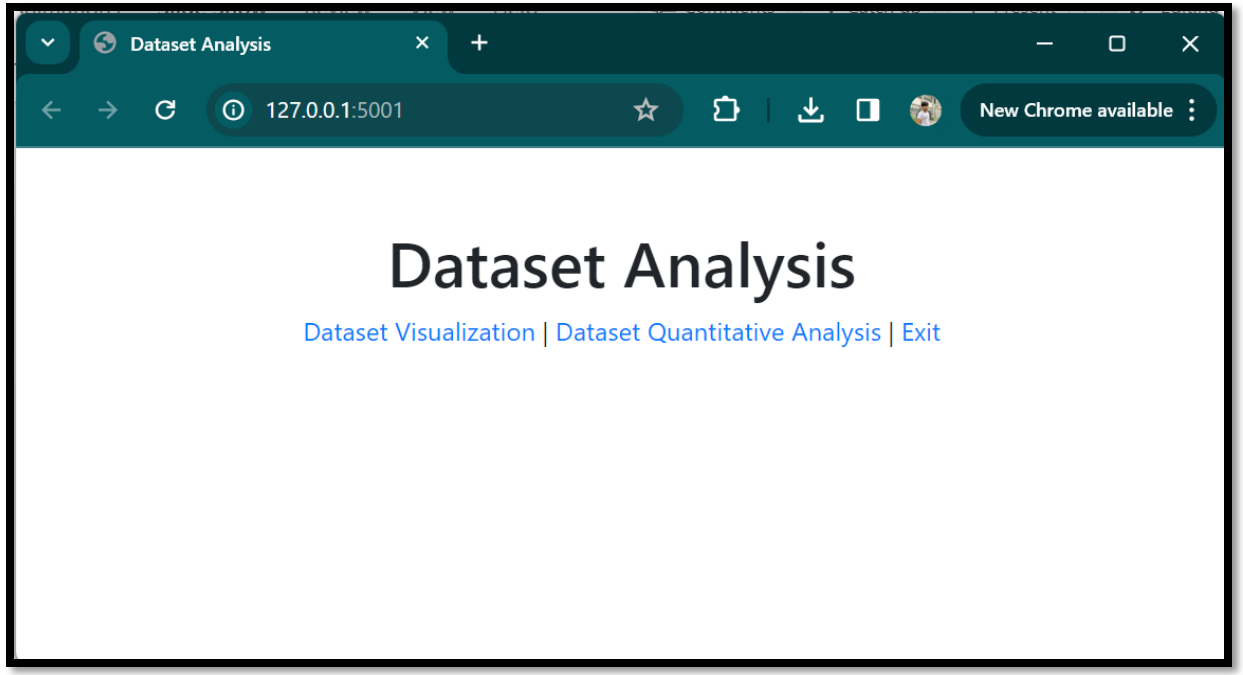
```
(minecpp) avulasaikrishna@hamilton:~/Sai_Work$ minecpp -u https://github.com/DenisCarriere/geocoder
[nltk_data] Downloading package punkt to
[nltk_data]     /home/avulasaikrishna/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Processing for project:  geocoder
Getting fixed commits...
Repo:  /home/avulasaikrishna/miniconda3/envs/minecpp/lib/python3.9/site-packages/Minecpp/geocoder/geocoder
Total comm:  Corresponding commit msg:  #108 Fixed Encoding with Google/Bing/ArcGIS
Processing   Processing Commits:  18%|     | 48/261 [05:11<39:55, 11.25s/it]Processing for commit:  9d1ce989b4c
Processing   02ac8162329369491ad64971fa5b1
0a89510b8b   Corresponding commit msg:  #107 Fix URL duplicate in Kwargs
Corresponding commit msg:  Import fix                                                    52d8463a634
Processing Commits:  99%|                                  | 259/261 [27:59<00:12,  6.35s/it]Processing for commit:  48bc43a6dc4
761eb7761b5b0affe2424a8ca6c84
Corresponding commit msg:  Merge pull request #342 from ThmsLa/master                    5e4860c876c
Processing Commits: 100%|                                  | 260/261 [27:59<00:04,  4.52s/it]Processing for commit:  39b9999ec70
e61da9fa52fe9fe82a261ad70fa8b
Corresponding commit msg:  Merge pull request #374 from vikaskyadav/patch-1              6833b13e355
Processing Commits: 100%|                                  | 261/261 [27:59<00:00,  6.46s/it]
Removing project folder:  geocoder
 * Serving Flask app 'Minecpp.app'                                                       533b8d7b24c
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

MineCPP: Flowchart



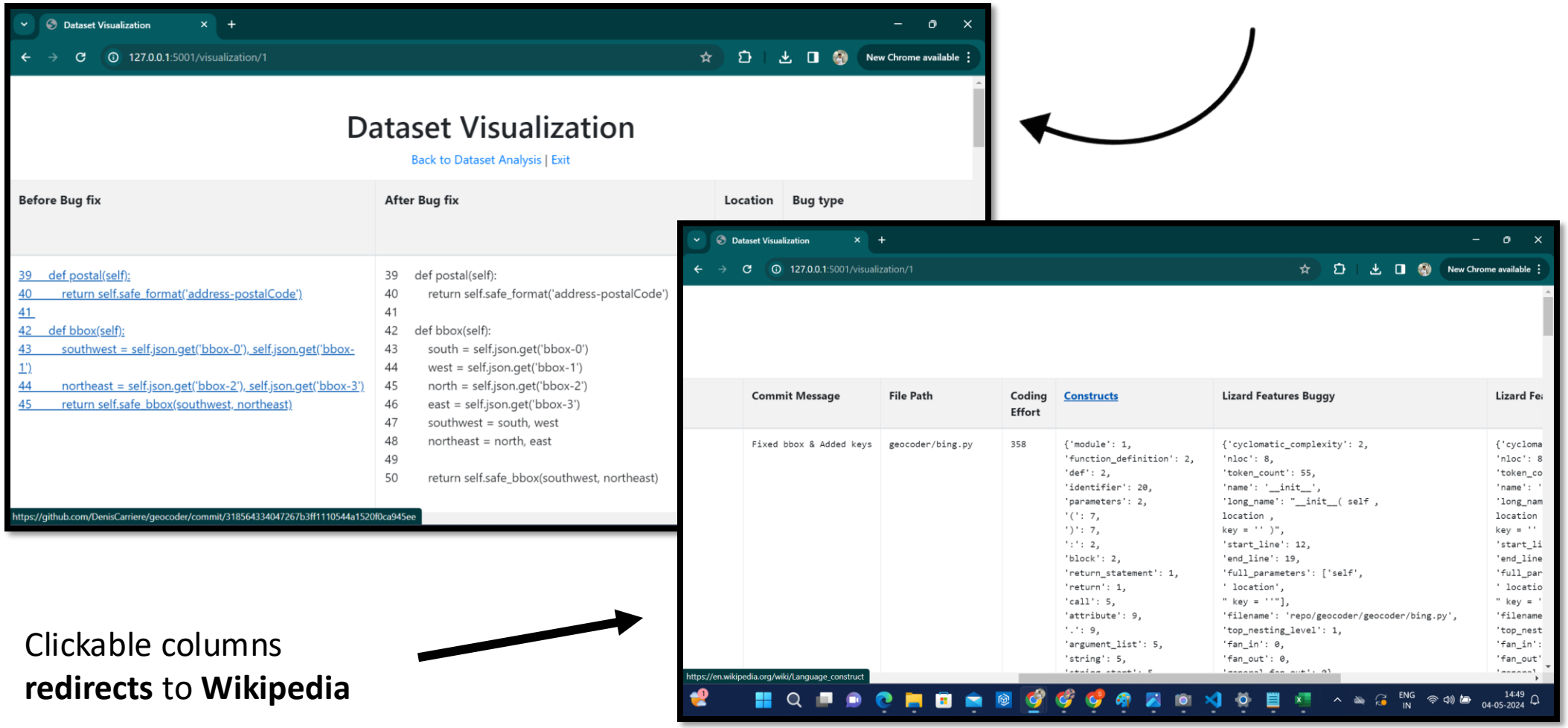
MineCPP: GUI (https://youtu.be/ln99irvbADE)

- Dataset Visualization
  - Dataset of bug-fix pairs is displayed on webpage
  - **Interactive** webpage for bug-fix pairs of a repository
- Quantitative analysis
  - **Coding Effort** vs **Bug-fix pairs**
  - **Similarity Score** vs **Bug-fix pairs**



Dataset Visualization

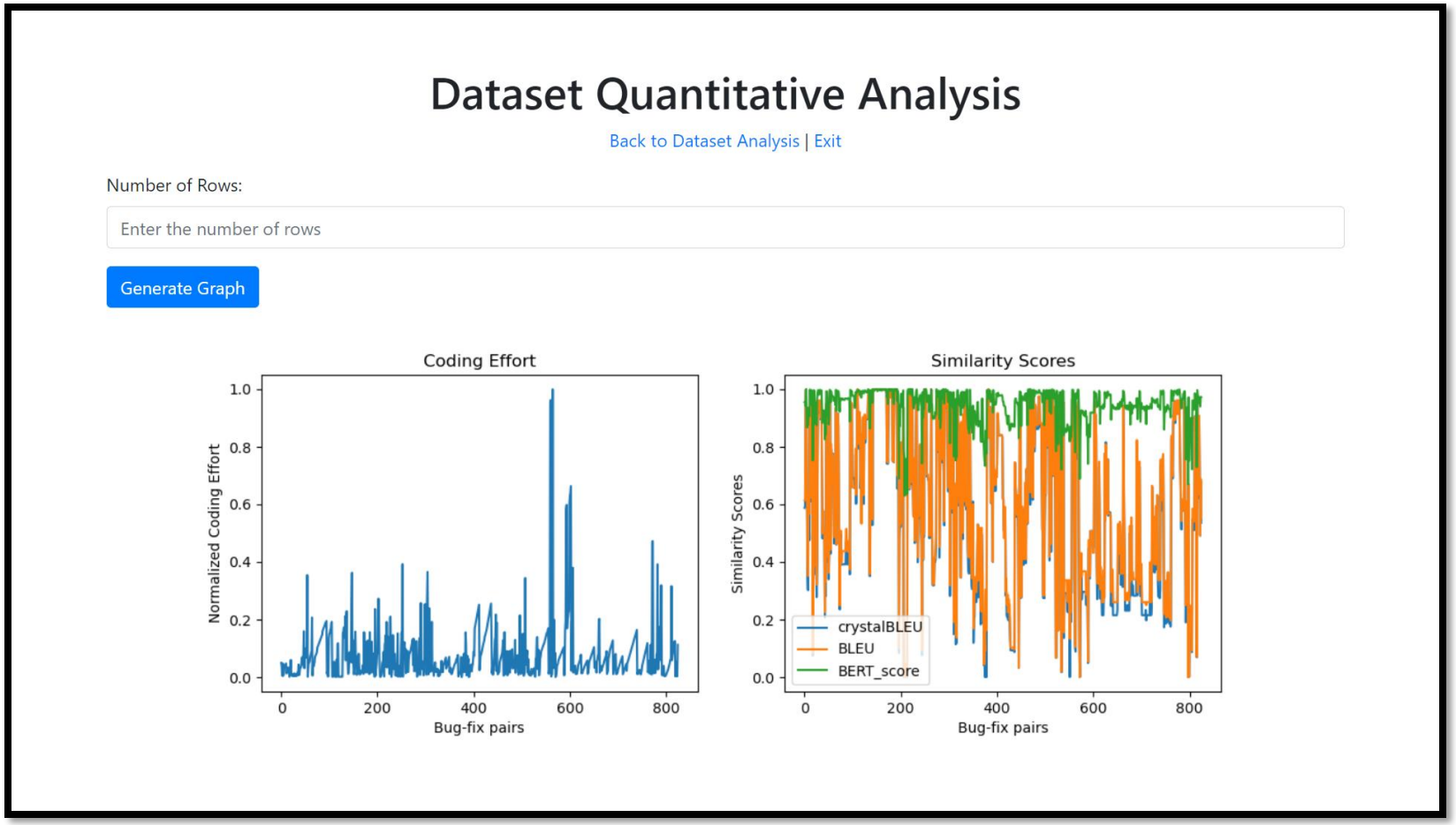
Clickable code snippets **redirects** to **GitHub source code**



Clickable columns **redirects** to **Wikipedia**

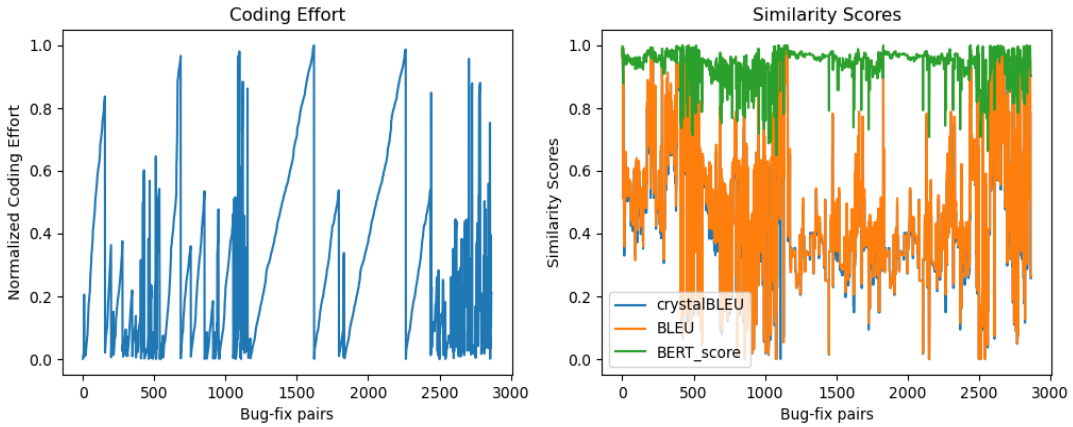
Dataset Quantitative Analysis

Example for Geocoder repository

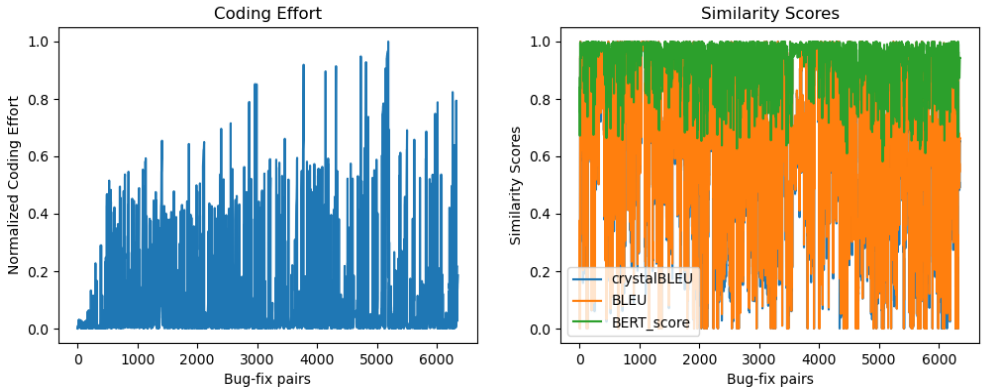


Evaluation

- MineCPP - run on **25** **Python** projects (**#stars**, **#commits**)
  - **Flask – 61243 Stars and 5096 commits**



  - **Yt-dlp– 35839 Stars and 21800 commits**



- **Sawtooth** pattern observed in **Coding Effort** vs **Bug-fix** pairs across projects
  - due to **modifications** in import statements or modifications at the **beginning** of the file.

Dataset: 25 Python projects

- **Schema** of the dataset

https://doi.org/10.5281/zenodo.10579446

Col#	Name	Type	Description
1	`Before Bug fix`	String	Code + Context of Bug
2	`After Bug fix`	String	Code + Context of after Bug fix
3	`Location`	List of ints.	Line #s of the bug and its fix
4	`Bug type`	String	Bug type deduced using `git diff` and LLM
5	`Commit Message`	String	Author's description about commit
6	`Project URL`	String	GitHub link of the repository
7	`File Path`	String	The path of file with a change or bug
8	`Fixed Commit`	String	git hash of `fixed` commit
9	`Buggy Commit`	String	git hash of `Buggy` commit
10	`Test File`	Bool	has test case or not
11	`Coding Effort`	Int	explained in the CodingEffort function
12	`Constructs`	Dict	type of constructs in which bug is present
13	`Lizard Features Buggy`	Dict	Cyclomatic Complexity of buggy code
14	`Lizard Features Fixed`	Dict	Cyclomatic Complexity of fixed code
15	`BLEU`	Float	text similarity score
16	`crystalBLEU_score`	Float	code similarity score
17	`BERT_score`	Float	code similarity score

Schema of dataset

MineCPP

*"Overall I think this might be a great tool if it worked as advertised. While the setup and tool execution were **really smooth**, also thanks to the **simple interface**, the obtained output made me wonder if the submitted paper only tells half the story. It seems that the obtained data may entail significant and non-trivial post-processing to obtain an actual bug dataset."*

-Reviewer (FSE)

Downstream Usage

- **Researchers and Academics:** can utilize the dataset for empirical studies, data driven bug detection and auto-fix algorithms, and pattern mining-based bug fix algorithms.
- **Software Developers:** can leverage the dataset/tool for enhancing their code quality and bug fixing processes.
- **Educators and Students:** can incorporate the dataset/tool into their software engineering courses or research projects.

Future Scope

- **Bug type**: development of a more generalized bug type classification system that transcends project-specific distinctions.
- **Accuracy**: application of static/dynamic analysis to identify and eliminate false positives.
- **User study**: manual/semi-automatic inspection by developers needed to validate our datasets

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 Sulistyo 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.