Mining Mutations from Public Python Projects

Adil Botabekov (20180806)
Nabila Sindi (20180744)
Nguyen Tien Dat (20190883)
Yehyoung Kang (20110162)

Problems

1. Mutation testing relies heavily on the quality of mutation operators
2. Common mutation operators may not be representative of actual bugs
3. It’s hard to come up with enough mutation operators manually

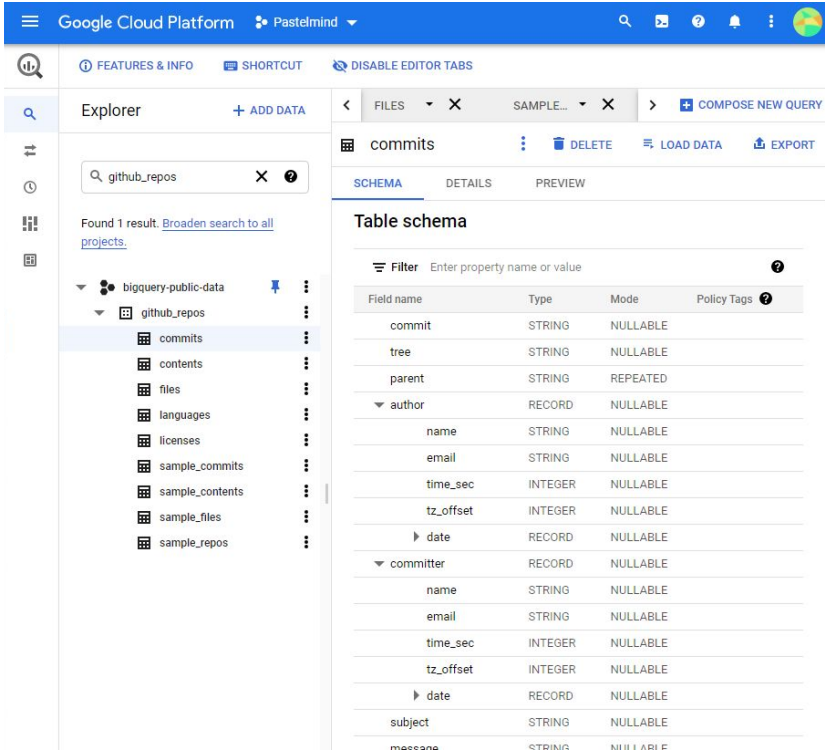
Solution: Automated mutation mining from real-world bugs

Collect commits with Google BigQuery

`bigquery-public-data.github_repos.commits`

- A public dataset mined from open-source GitHub repositories
- 870GB table containing 250 million commits

We needed **bug fixes** for **Python** code



Collect commits with Google BigQuery

💡 Observation:

- By convention, developers use certain words in commit messages that fix bugs
- e.g. “This fixes #23”

We selected commits that...

- Change > 1 Python file
- The message contains a keyword often used when fixing a commit (bug, fix, issue, error)

```
SELECT commit, subject, repo_name, ARRAY(
  SELECT AS STRUCT *
  FROM UNNEST(difference)
  WHERE (new_path LIKE "%.py")
) AS difference
FROM `bigquery-public-data.github_repos.sample_commits`
WHERE EXISTS (
  SELECT new_path FROM UNNEST(difference)
  WHERE (new_path LIKE "%.py")
)
AND regexp_contains(subject, 'bug|fix|issue|error')
```

Collect commits with Google BigQuery

End result: 3.3 million commits with

- Repository name
- SHA-1 hash of the commit
- SHA-1 hash of the parent commit
- *Paths* of all Python files changed

mined_data_2

QUERY SHARE COPY

SCHEMA DETAILS PREVIEW

200 row per page limit reached due to duplicate values or complex results. Displaying 53 results to reflect this.

44	5abe910b324d9caa125068065974958a734c300d	9dd54c1f2ce0d3010208e176caa0dc4e7df086aa	fixed untracked files
45	e2f5fe6502e9ba0be4aed90bca997766e9753527	d2da252a0a83a6c8ebcc53eeea1f2d65b417a43b	Fixed bug with traps. Updated project struc
46	b5c4daa2b0abe96583a67796590974cecce54370	87cb11ac1584c4963798e1391833ee0b29512ada	Delete error_testing.py
47	18dd8228b2fa8463f12eed765adf9cc914415cda	557660a622ac9edc6cb326ee93fa954c7c422ed9	Regression: review fix
48	6ad65342cd4808505ad0512a5b3b12d693f6b64c	82d41cb12a6509482ec1ed470462c43026401a12	Revert "fix it for maven script with copy sky-
49	6f18a819e79d031aca4fdfec3e57be606f694f65	feb25bed5887922ca212dfe696e3b0595d3266d3	[Dee] Build fix
50	c276375b4d6cad681d45f12a62d463a194e356c8	7bbb540b8822306a361b06f34c2ceb4b8c8351cb	Removing old fixtures to a submodule.
51	6f6473f1eae1b8a9186d8b13c6f1cc55f18d84fe	1b5738e3f50a84cf0e4142ced53bfc3dd16654a7	fixing diurectory name
52	8faa76d4c2a7a7aaf5f1974f16bd58308cdd765e	fec301e8d8bd2071561e23fbb5aad1071d73f41b	remove generate enum python script. fixes
53	a9b743a38fc7f7235a98f14dcea5899879af06bf	f6853629a544f81c84b4ae389a3f376bbfcb0472	trying to fix svn eror

Rows per page: 53 1 - 53 of 3341289

```python
class GithubFilesSpider(scrapy.Spider):
    name = f"github_files_chunk{CHUNK_NUMBER}"
    allowed_domains = ["raw.githubusercontent.com"]

    CHUNK_FILE_NAME = CHUNK_FILE_NAME
    OUTPUT_FILE_PATH = OUTPUT_FILE_PATH

    def start_requests(self):
        # We intentionally read the file twice instead of storing the file in
        # a list. This is because the chunk files eat up a large amount of
        # memory (~300 MB observed).
        ROW_COUNT = sum(1 for _ in load_gzipped_lines(file_name=self.CHUNK_FILE_NAME))

        # For resuming interrupted crawling sessions
        downloaded_changes = load_downloaded_changes(file_path=self.OUTPUT_FILE_PATH)
        logging.info(
            f"Found {len(downloaded_changes)} change entries that were downloaded in a previous run"
        )

        skipped_file_count = 0

        for row_index, row in enumerate(
            load_gzipped_lines(file_name=self.CHUNK_FILE_NAME)
        ):
            file_change_entries = list(
                generate_file_change_data(row=row, row_index=row_index)
            )

            for fc_num, fc_data in enumerate(file_change_entries, start=1):
                if is_already_downloaded(downloaded_changes, fc_data):
                    logging.debug(
                        f"Skipping already downloaded commit {row_index + 1} / {ROW_COUNT}, file {fc_num} / {len(file_change_entries)}"
                    )
                    skipped_file_count += 1
                    if skipped_file_count % 1000 == 0:
                        logging.info(
                            f"Skipped {skipped_file_count} files that were already downloaded"
                        )
                    continue
```

Crawling GitHub with Scrapy

Use Scrapy to download commits directly from GitHub

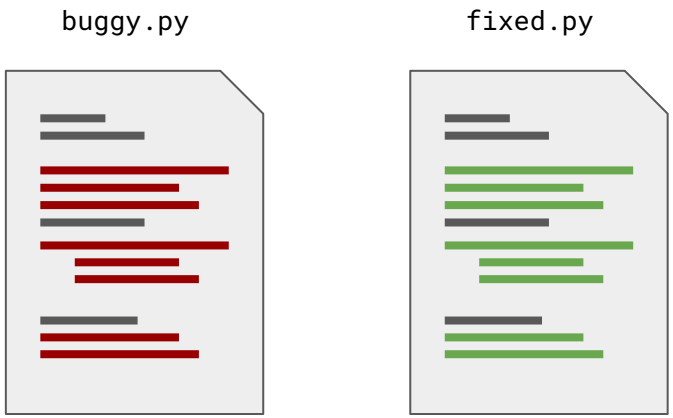
Downloaded 9.3 million Python files (200GB) over 3 weeks

(GitHub limits server hits to 5000/hour 😡)

https://scrapy.org/

Preprocessing: Pairing

A file may contain changed & unchanged code regions

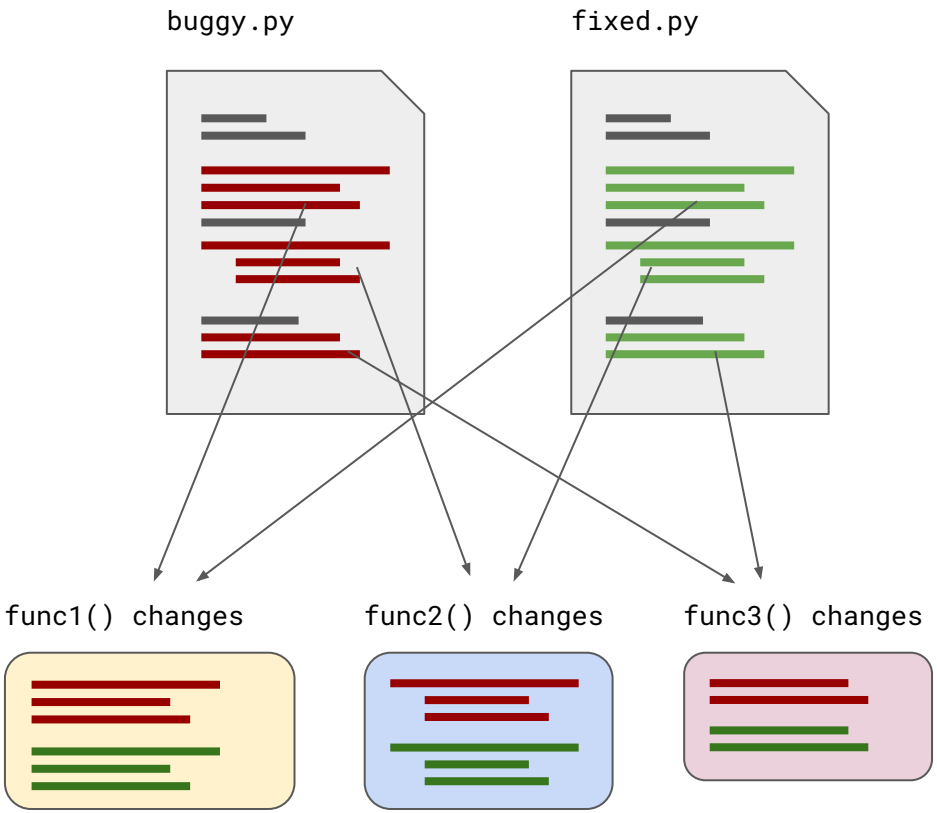


Preprocessing: Pairing

A file may contain changed & unchanged code regions

⇒ Extract **function-level changes**

- Pick functions and methods
- Find matching pairs of functions by name
  - Python functions are rarely overloaded, so function names are *usually* unique
- Ignore function pairs that are unchanged



Preprocessing: Extraction

How to describe changes in code?

⇒ GumTreeDiff, an AST comparison tool that generates a sequence of *edit actions*

```
-self.discovered_filesystems = []
+self.discovered_filesystems = set()
```

```
[
  "insert-node",
  "insert-node",
  "insert-node",
  "delete-node"
]
```

https://github.com/GumTreeDiff/gumtree

Preprocessing: Normalization

Python code contains many distinct values and identifiers

- Large “vocabulary” of words, hard to generalize

⇒ Replace variables and constants with placeholders

⇒ Preserve frequently used “idioms”: `x`, `i`, `print`, …

```python
def add_details(self, info):
    for (k, v) in info.items():
        try:
            setattr(self, k, v)
        except AttributeError:
            pass
```

→

```python
def IDENTIFIER_0(self, IDENTIFIER_1):
    for (k, v) in IDENTIFIER_1.IDENTIFIER_2():
        try:
            IDENTIFIER_3(self, k, v)
        except IDENTIFIER_4:
            pass
```

Preprocessing: Filtering

Many code changes are too large for analysis

⇒ Select functions that have ≤ 50 lexer tokens per function

- We wanted to train an RNN model with seq2seq, which accepts 50 “words” per sentence
  - Unfortunately, we didn’t have enough time to actually use RNN…

⇒ Processed and filtered 475,961 function pairs

Data format

1. `data.jsonl`

Fields of each dictionary:

- `name`
- `before_code`: source code before bugfix
- `after_code`: source code after bugfix
- `before_code_normalized`, `after_code_normalized`: Abstracted & normalized code
- edit_actions: List of AST edit actions
- replacement_map


```jsonl
{
  "name": "intel-hpdd/intel-manager-for-lustre:ff337f5ac809a9dbf2cefb988e595fad696acf8a:chroma-manager/chroma_core/lib/detection.py:DetectScan.__init__",
  "before_code": "def __init__(self, step):\n    self.created_filesystems = []...",
  "after_code": "def __init__(self, step):\n    self.created_filesystems = []...",
  "before_code_normalized": "def __init__ ( self , IDENTIFIER_0 ) : ...",
  "after_code_normalized": "def __init__ ( self , IDENTIFIER_0 ) : ...",
  "edit_actions": [
    "insert-node",
    "insert-node",
    "insert-node",
    "delete-node"
  ],
  "replacement_map": {
    "identifiers": {
      "IDENTIFIER_0": "step",
      "IDENTIFIER_1": "created_filesystems",
      "IDENTIFIER_2": "discovered_filesystems",
      "IDENTIFIER_3": "created_mgts"
    },
    "floats": {},
    "ints": {},
    "strings": {},
    "f_strings": {}
  }
}
```

Data format

2. `buggy.txt`

Each line is abstracted buggy code

3. `fixed.txt`

Each line is abstracted fixed code

(whitespace replaced with special tokens)

buggy.txt

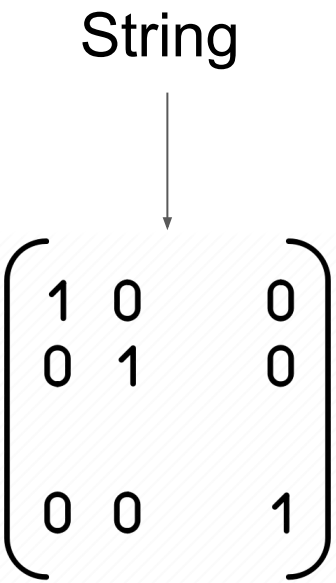
```
1 def IDENTIFIER_0 ( self ) : $NEWLINE $INDENT super ( IDENTIFIER_1 , self ) .
  IDENTIFIER_0 ( ) $NEWLINE IDENTIFIER_2 ( quiet = True ) $NEWLINE assert
  self . client . IDENTIFIER_3 ( username = STR_0 , password = 'test' )
  $NEWLINE $DEDENT
2 def IDENTIFIER_0 ( self ) : $NEWLINE $INDENT super ( IDENTIFIER_1 , self ) .
  IDENTIFIER_0 ( ) $NEWLINE IDENTIFIER_2 ( quiet = True ) $NEWLINE assert
  self . client . IDENTIFIER_3 ( username = STR_0 , password = 'test' )
  $NEWLINE $DEDENT
3 def IDENTIFIER_0 ( self ) : $NEWLINE $INDENT super ( IDENTIFIER_1 , self ) .
  IDENTIFIER_0 ( ) $NEWLINE IDENTIFIER_2 ( quiet = True ) $NEWLINE assert
  self . client . IDENTIFIER_3 ( username = STR_0 , password = 'test' )
  $NEWLINE $DEDENT
```

fixed.txt

```
1 def IDENTIFIER_0 ( self ) : $NEWLINE $INDENT super ( IDENTIFIER_1 , self ) .
  IDENTIFIER_0 ( ) $NEWLINE assert self . client . IDENTIFIER_3 ( username =
  STR_0 , password = 'test' ) $NEWLINE $DEDENT
2 def IDENTIFIER_0 ( self ) : $NEWLINE $INDENT super ( IDENTIFIER_1 , self ) .
  IDENTIFIER_0 ( ) $NEWLINE assert self . client . IDENTIFIER_3 ( username =
  STR_0 , password = 'test' ) $NEWLINE $DEDENT
3 def IDENTIFIER_0 ( self ) : $NEWLINE $INDENT super ( IDENTIFIER_1 , self ) .
  IDENTIFIER_0 ( ) $NEWLINE assert self . client . IDENTIFIER_3 ( username =
  STR_0 , password = 'test' ) $NEWLINE $DEDENT
```

Vectorization / Embedding

1. Edit actions between buggy and fixed code are tokens like [delete-node], [move-tree], etc.
2. Train Gensim's Doc2Vec model on sequence of edit actions.
3. Doc2Vec produces vectors of fixed size such that similar sequences of edit actions can be calculated to be close (e.g. using cosine distance)...hopefully



Clustering algorithms

Several options:

K-Means

Why?

A simple and powerful classic go-to

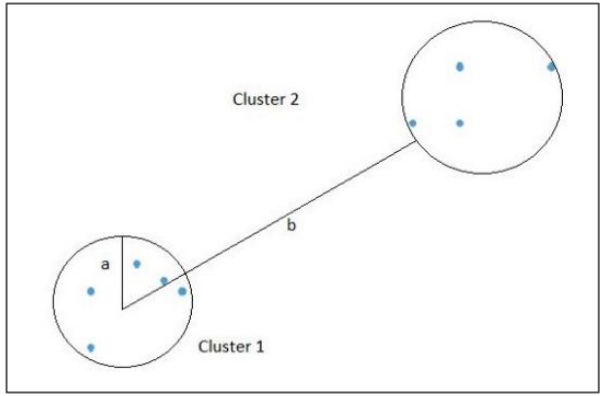
Gaussian Mixture

Why?

As opposed to K-Means, utilizes the variance of the data and not just the mean.

Cluster evaluation (Silhouette score)

- A metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1
- 1 means clusters are well apart from each other and **clearly distinguished**.
- 0 means clusters are indifferent, or in other word, the distance between clusters is **not significant**
- -1 means clusters are assigned in the **wrong way**



Silhouette Score =
(b-a)/max(a,b)

a= average intra-cluster distance
b= average inter-cluster distance

Cluster evaluation (Silhouette score)

For vector size = 5:

Number of clusters	Silhouette score
2	0.36445
3	**0.42311**
4	0.38316
5	0.34782
6	0.35342

For vector size = 20:

Number of clusters	Silhouette score
2	0.35168
3	0.41482
4	0.39859
5	0.37131
6	0.34029

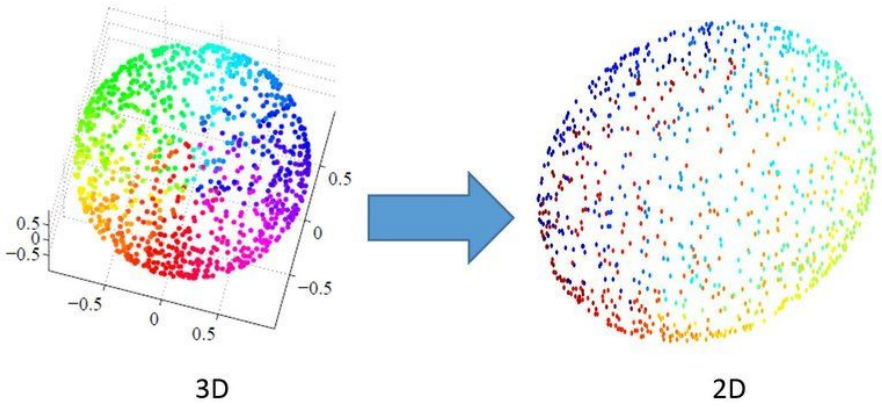
For vector size = 40:

Number of clusters	Silhouette score
2	0.35446
3	0.41928
4	0.40131
5	0.37247
6	0.35874

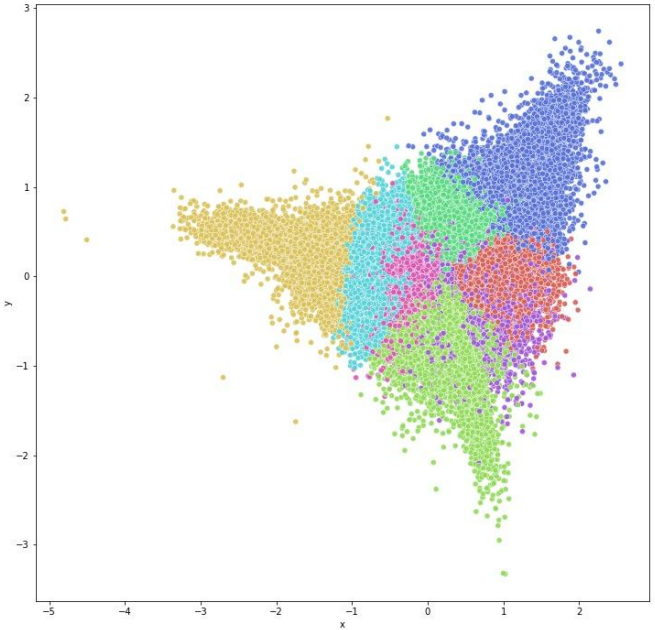
Clustering visualized

Data is not in 2D! How do we visualize?

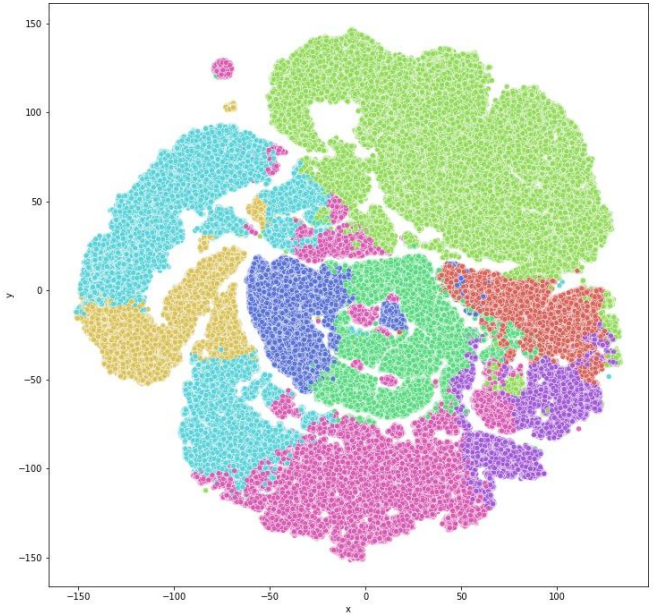
Use t-SNE and PCA algorithms for **dimensionality reduction**.



t-SNE + PCA > PCA

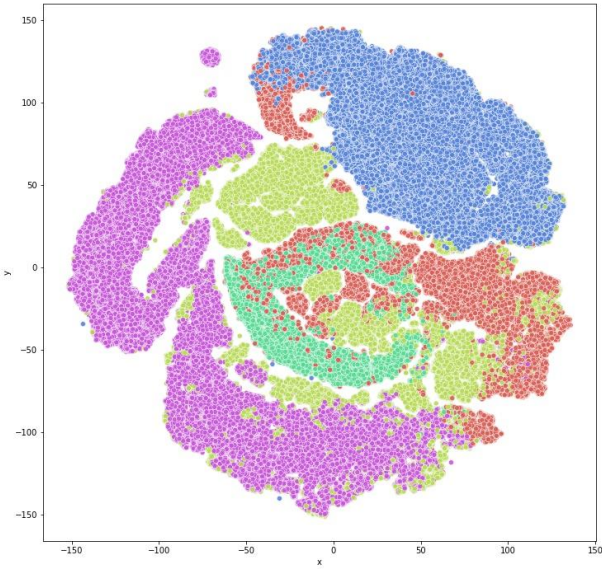


PCA | K-Means 8 Clusters

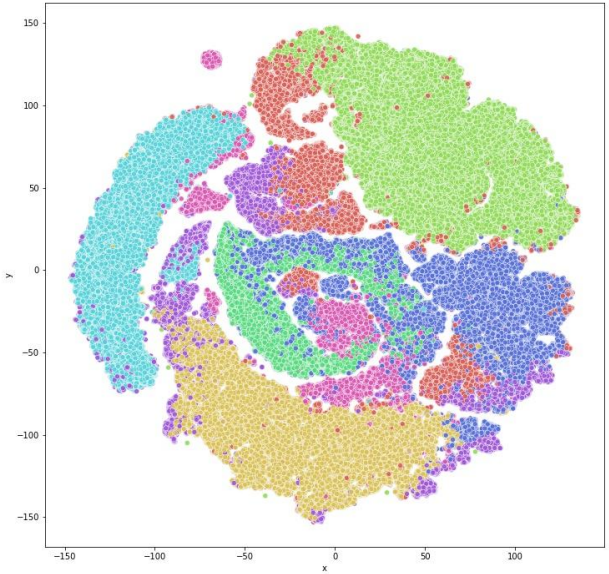


PCA + t-SNE | K-Means 8 Clusters

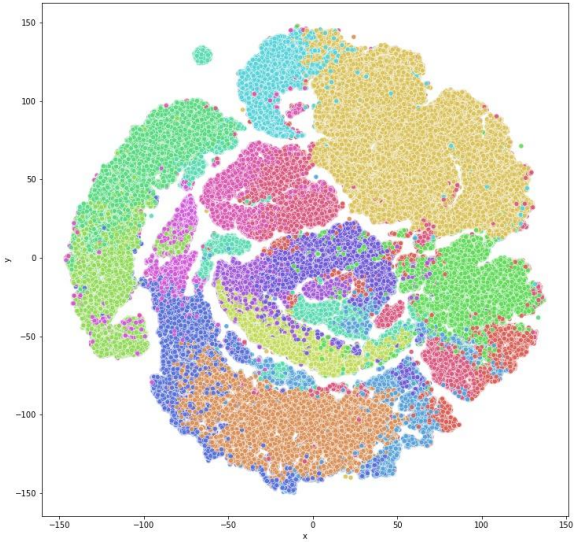
Clusters | Gaussian Mixture Model



5 clusters

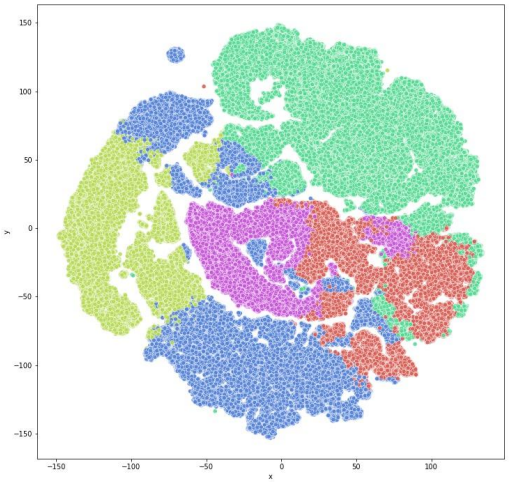


8 clusters

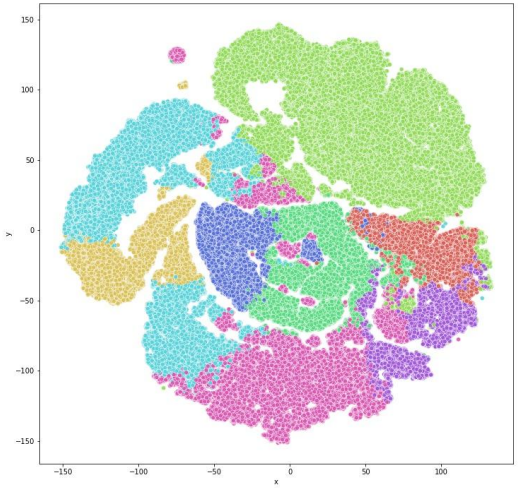


16 clusters

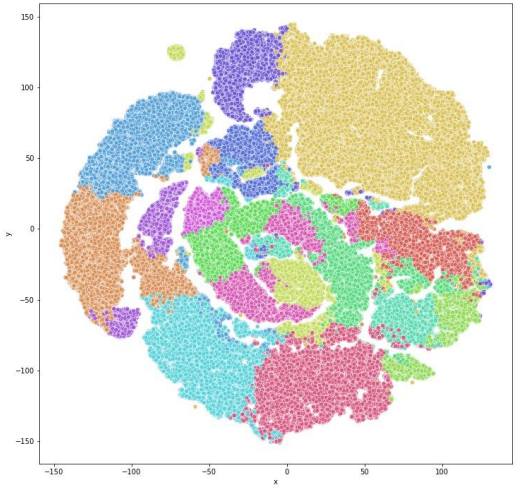
Clusters | K-Means



5 clusters

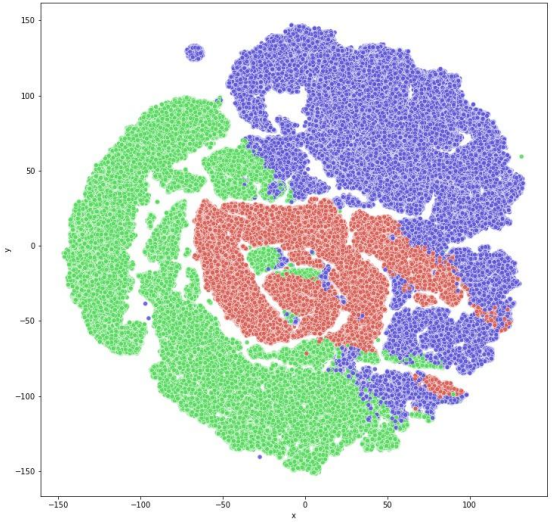


8 clusters

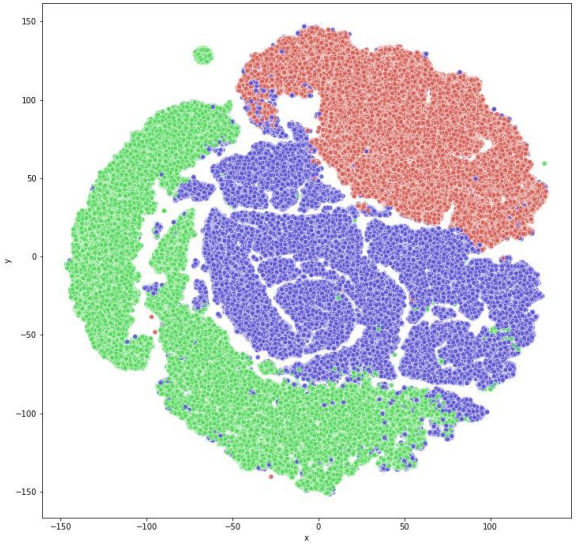


16 clusters

K = 3, Best Silhouette score



K-Means



Gaussian Mixture

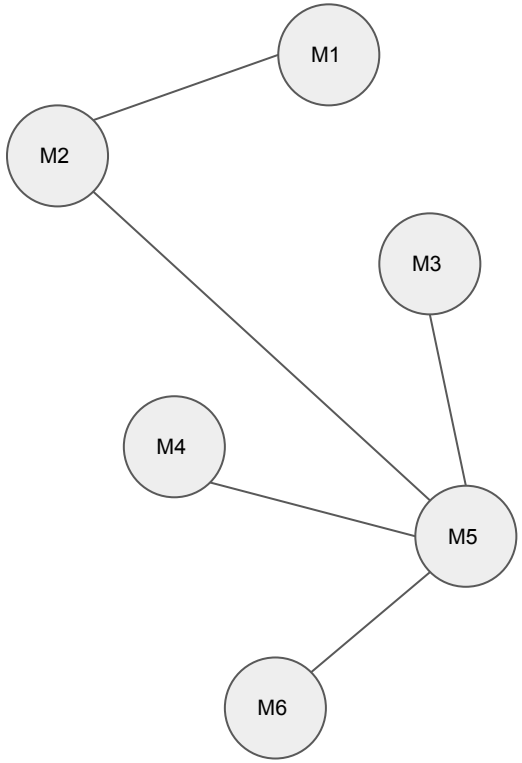
Sampling algorithm

For each cluster, build a graph. Each “mutation” is a node.

How do we add edges?

1. We calculate the cosine distance between all pairs of points in the set of embeddings

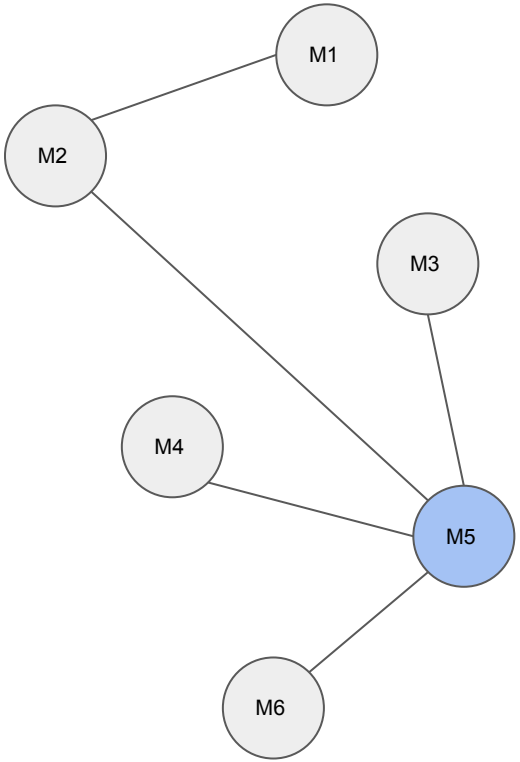
2. For pairs with cosine distance larger than a certain threshold, the edge is added



Sampling algorithm (cont.)

Two ways to rank nodes:

1. Choose one with the largest degree (no. of edges)
2. **Use Page Rank algorithm**



Samples | ["update-node"]

BEFORE

```
def IDENTIFIER_0 ( self ) :
    self. IDENTIFIER_1( )
    IDENTIFIER_3 . IDENTIFIER_2 ( STR_0 )
    print ( IDENTIFIER_4 ( IDENTIFIER_6 . IDENTIFIER_5 () ) )
```

AFTER

```
def IDENTIFIER_0 ( self ) :
    self . IDENTIFIER_7 ( )
    IDENTIFIER_3 . IDENTIFIER_2 ( STR_0 )
    print ( IDENTIFIER_4 ( IDENTIFIER_6 . IDENTIFIER_5 () ) )
```

Samples | ["insert-node", "delete-node"]

BEFORE

```
def IDENTIFIER_0 ( IDENTIFIER_1 , IDENTIFIER_2 ) :
    f = c . IDENTIFIER_3 ( STR_0 % ( IDENTIFIER_1 , IDENTIFIER_2 ) )
    return f is not 0
```

AFTER

```
def IDENTIFIER_0 ( IDENTIFIER_1 , IDENTIFIER_2 ) :
    f = c . IDENTIFIER_3 ( STR_0 % ( IDENTIFIER_1 , IDENTIFIER_2 ) )
    return f != 0
```

Samples | ["insert-node", "insert-node", "move-tree"]

BEFORE

```
def IDENTIFIER_0 ( self , IDENTIFIER_1 ) :
    x = IDENTIFIER_1 . IDENTIFIER_2 ( int , str )
    if IDENTIFIER_3 ( x ) is int :
        x = IDENTIFIER_4 [ x ]
    IDENTIFIER_1 . IDENTIFIER_5 ( x )
```

AFTER

```
def IDENTIFIER_0 ( self , IDENTIFIER_1 ) :
    x = IDENTIFIER_1 . IDENTIFIER_2 ( int , str )
    if IDENTIFIER_3 ( x ) is int :
        x = IDENTIFIER_1 . IDENTIFIER_4 [ x ]
    IDENTIFIER_1 . IDENTIFIER_5 ( x )
```

Samples | ["delete-node"]

BEFORE

```
def IDENTIFIER_0 ( self , name , value ) :
    if value == None :
        value = ''
    value = str ( value )
    self . IDENTIFIER_1 . append ( ( name , value ) )
    return
```

AFTER

```
def IDENTIFIER_0 ( self , name , value ) :
    if value is None :
        value = ''
    value = str ( value )
    self . IDENTIFIER_1 . append ( ( name , value ) )
    return
```

Current Difficulties or Limitations

1. Hard to automate evaluation of whether these clustering and sampling techniques produce useful mutations

2. Hard to evaluate usefulness of mutations (could be subjective)

3. Hyperparameters for dimensionality reduction, clustering, sampling and embedding model are not optimized

Possible Improvements

1. Utilize GumTree better

2. Utilize normalized before and after code. (Utilize cosine distance between the embedding of before and the embedding of the after code)

3. Tune hyperparameters

4. Train an ML model that given fixed code as input, the model can learn to generate buggy code as output.

Conclusion

1. The technique is valid and has potential

2. With the aforementioned improvements, we’re confident that a powerful neural mutation testing tool is possible

3. It will allow for world domination :)

Summary

1. **Problem**: Mutation testing tools offer a very limited set of mutation operators

2. **Approach**: Mine bug-fixes from public GitHub commits, group them into semantically close groups and choose the best samples from each group

3. **Results**: Valid mutations could be found, although many of them are not too different from regular mutation operators. The ones that are different may or may not be hard to implement.

QUESTIONS?