Assignment-1

Software Tools and Techniques for CSE

Shardul Junagade (23110297)
Repository: cs202-stt

September 8, 2025

Contents

Lab 3: Multi-Metric Bug Context Analysis and Agreement Detection in Bug-Fix Commits

Repository Link: cs202-stt/lab3

1.1 Introduction, Setup, and Tools

1.1.1 Introduction

In Lab 2, I had prepared a per-file dataset of bug-fix commits with extracted diffs and model-generated summaries. The purpose of this lab was to analyse the relation between structural code quality and magnitude of changes in bug-fix commits. Specifically, I aimed to investigate:

- Structural metrics around each fix (Maintainability Index, Cyclomatic Complexity, and Lines of Code) using radon.
- Change magnitude metrics (Semantic similarity with CodeBERT and Token similarity with BLEU).
- Classify each fix as Major or Minor from both lenses and check where they agree (or don't).

By combining these, I classified the bug fixes as Major or Minor and checked where the structural and semantic lenses agreed or conflicted. This is important because commit messages or diffs alone rarely capture how "big" or "complex" a change really is.

1.1.2 Environment and Tools

- OS: Windows 11, Terminal: PowerShell 7
- Code Editor: Visual Studio Code - Insiders
- Python: 3.13.7
- Key packages: radon, nltk, transformers, torch, scikit-learn
- Models: microsoft/codebert-base (for embeddings)
- Hardware: NVIDIA RTX 4060 Laptop GPU

```
Python version: 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)]
Using device: cuda
PyTorch version: 2.8.0+cu129
CUDA version: 12.9
Device name: NVIDIA GeForce RTX 4060 Laptop GPU

radon: 6.0.1
transformers: 4.56.0
scikit-learn: 1.7.1
numpy: 2.3.2
pandas: 2.3.2
tqdm: 4.67.1
nltk: 3.9.1
```

Environment Setup

1.2 Methodology and Execution

Notebook Link: lab3.ipynb

1.2.1 Starting point (Lab 2 dataset)

- Input CSV: `lab3/lab2_diffs.csv`
- Columns included: Commit Hash, Message, File Name, Source Code (before), Source Code (current), Diff, LLM Inference (fix type), Rectified Message

I first loaded the dataset, checked the first 10–20 rows, and verified that there were no missing critical columns. Then, I checked for NaNs in key columns in the dataset.

Hash	Message	File Name	File Path	Change Type	Source Code (before)	Source Code (current)	Diff	LLM Inference (fix type)	Rectified Message
ɔc6e6de65a86	A few fixes to initial point_thickness impleme...	constants.py	constants.py	ModificationType.MODIFY	import os\nimport numpy as np\n\n\nGENERALLY_B...	import os\nimport numpy as np\n\nPRODUCTION_QU...	@@ -1,9 +1,6 @@\n import os\n import numpy as ...	add missing constants	[constants.py]: Fix default point thickness to...
ɔc6e6de65a86	A few fixes to initial point_thickness impleme...	displayer.py	displayer.py	ModificationType.MODIFY	import numpy as np\nimport itertools as it\nim...	import numpy as np\nimport itertools as it\nim...	@@ -55,7 +55,7 @@ def paint_mobjects(mobjects,...	add nudge to displayer.py	[displayer.py]: Fix potential offset issue in ...
ɔc6e6de65a86	A few fixes to initial point_thickness impleme...	mobject.py	mobject\mobject.py	ModificationType.MODIFY	import numpy as np\nimport itertools as it\nim...	import numpy as np\nimport itertools as it\nim...	@@ -21,7 +21,7 @@ class Mobject(object):\n ...	add missing docstring	[mobject.py]: Fix inconsistent point_thickness...
)ed28d95ad82	middle of massive restructure, everything stil...	__init__.py	__init__.py	ModificationType.MODIFY	from animation import *\nfrom mobject import *...	from animation import *\nfrom scene import *\n...	@@ -1,10 +1,13 @@\n from animation import *\n-...	add missing import statements	[__init__.py]: Fix [bug/issue/etc] in [functio...
)ed28d95ad82	middle of massive restructure, everything stil...	__init__.py	animation__init__.py	ModificationType.MODIFY	from animation import *\nfrom transform import...	from animation import *\nfrom meta_animations ...	@@ -1,3 +1,4 @@\n from animation import *\n-fr...	add missing newline	Fix regression in transform module by adding m...
)ed28d95ad82	middle of massive restructure, everything stil...	animation.py	animation\animation.py	ModificationType.MODIFY	from PIL import Image\nfrom colour import Colo...	from PIL import Image\nfrom colour import Colo...	@@ -7,11 +7,10 @@ import os\n import copy\n im...	add missing config to missing color animation	[animation.py]: Fix bug in writeGif function b...
)ed28d95ad82	middle of massive restructure, everything stil...	meta_animations.py	animation\meta_animations.py	ModificationType.ADD	NaN	import numpy as np\nimport itertools as it\nfr...	@@ -0,0 +1,95 @@\n+import numpy as np\n+import...	add tests for animation.update and animation.u...	[meta_animations.py]: Fix bug in DelayByOrder ...
)ed28d95ad82	middle of massive restructure, everything stil...	simple_animations.py	animation\simple_animations.py	ModificationType.MODIFY	import numpy as np\nimport itertools as it\nfr...	import numpy as np\nimport itertools as it\nfr...	@@ -2,42 +2,12 @@ import numpy as np\n import ...	add some more classes to the animation class	[simple_animations.py]: Refactor commit messag...
)ed28d95ad82	middle of massive restructure, everything stil...	transform.py	animation\transform.py	ModificationType.MODIFY	import numpy as np\nimport itertools as it\nim...	import numpy as np\nimport itertools as it\nim...	@@ -4,39 +4,12 @@ import inspect\n import copy...	add new animation classes	[transform.py]: Fix broken behavior in path_al...

Dataset Preview

```
# check for nan values in the df
print("Checking for NaN values in each column:")
print(df.isna().sum())
```

```
Checking for NaN values in each column:
Hash                        0
Message                     0
File Name                   0
File Path                   0
Change Type                 0
Source Code (before)       52
Source Code (current)      35
Diff                        0
LLM Inference (fix type)    0
Rectified Message           0
dtype: int64
```

NaN Counts

1.2.2 Baseline Descriptive Stats

From the dataset, I computed the following statistics to understand the dataset:

- Total unique commits and total file entries.
- Average modified files per commit.
- Distribution of fix types from `LLM Inference (fix type)`.
- Most frequently modified filenames and extensions.

The following images show the code and output for these computations:

```python
total_commits = df['Hash'].nunique()        # total number of unique commits
total_files = df.shape[0]                   # total number files
avg_files_per_commit = total_files / total_commits      # average number of files per commit
# avg_files_per_commit_pd = df.groupby('Hash').size().mean()

# Distribution of fix types
fix_type_distribution = df["LLM Inference (fix type)"].value_counts(dropna=False)

# Most frequently modified filenames and extensions
most_modified_filenames = df['File Name'].value_counts()
df['file_extension'] = df['File Name'].apply(
    lambda x: '.' + x.split('.')[-1] if pd.notna(x) and '.' in x else 'no_extension'
)
file_extension_distribution = df['file_extension'].value_counts()


print(f"Total number of unique commits: {total_commits}")
print(f"Total number of files: {total_files}")
print(f"Average number of modified files per commit (manual calc): {avg_files_per_commit:.2f}")
# print(f"Average number of modified files per commit (pandas calc): {avg_files_per_commit_pd:.2f}")
print("\nDistribution of fix types:")
print(fix_type_distribution.head(10))
print("\nMost frequently modified filenames:")
print(most_modified_filenames.head(10))
print("\nDistribution of file extensions:")
print(file_extension_distribution.head(10))
```

Baseline Stats Code

```
Total number of unique commits: 1121
Total number of files: 2041
Average number of modified files per commit (manual calc): 1.82

Distribution of fix types:
LLM Inference (fix type)
add missing docstring          161
add missing import              95
add missing imports             39
add missing docstrings          29
add missing class attributes    28
add missing comments            27
add missing comment             18
add missing config file         17
update camera.py                15
add missing config              13
Name: count, dtype: int64
```

Baseline Stats

```
Most frequently modified filenames:
File Name
vectorized_mobject.py    120
mobject.py               113
scene.py                  87
geometry.py               83
camera.py                 60
svg_mobject.py            56
tex_mobject.py            47
config.py                 44
constants.py              36
coordinate_systems.py     36
Name: count, dtype: int64

Distribution of file extensions:
file_extension
.py           1809
.glsl           78
.rst            47
.yml            34
.md             27
.txt             9
.mp4             8
.svg             7
.gitignore       6
.tex             6
Name: count, dtype: int64
```

Baseline Stats

1.2.3 Structural metrics with radon

For each file, I ran radon on both "before" and "current" versions of the source code and recorded the following structural metrics:

- **Maintainability Index (MI)** – A composite score that combines factors like lines of code, complexity, and comments to indicate how easy a piece of code is to maintain. A higher MI usually means the code is more readable and maintainable.
- **Cyclomatic Complexity (CC)** – A measure of how many independent paths exist through the code, essentially capturing the decision points (like if/else, loops). Higher CC means the code is more complex and harder to test thoroughly.
- **Lines of Code (LOC)** – The raw number of lines in the code. While simple, this metric is a direct measure of the size of the code and often correlates with the effort required to understand or modify it.

I then computed their deltas: `MI_Change`, `CC_Change`, `LOC_Change`. I caught any parsing exceptions and recorded them as NaN (those propagate to "Unknown" later). I saved these results to `results/structural_metrics.csv`.

The following image show the code implementation for structural metrics computation:

```python
def compute_radon_metrics(code):
    """Compute radon metrics: MI, CC, LOC for given source code."""
    if not isinstance(code, str) or code.strip() == "":
        return (np.nan, np.nan, np.nan)

    try:
        # Maintainability Index
        mi = float(mi_visit(code, multi=True))
    except Exception as e:
        mi = np.nan

    try:
        # Cyclomatic Complexity
        cc_results = cc_visit(code)
        cc = np.mean([block.complexity for block in cc_results]) if cc_results else 0.0
    except Exception as e:
        cc = np.nan

    try:
        # Lines of Code
        loc = radon_analyze(code).loc
    except Exception as e:
        loc = np.nan

    return (mi, cc, loc)


# Compute radon metrics for each file before and after the commit
df[['MI_Before', 'CC_Before', 'LOC_Before']] = df['Source Code (before)'].progress_apply(
    lambda x: pd.Series(compute_radon_metrics(x))
)
df[['MI_After', 'CC_After', 'LOC_After']] = df['Source Code (current)'].progress_apply(
    lambda x: pd.Series(compute_radon_metrics(x))
)

# Compute changes in metrics
df['MI_Change'] = df['MI_After'] - df['MI_Before']
df['CC_Change'] = df['CC_After'] - df['CC_Before']
df['LOC_Change'] = df['LOC_After'] - df['LOC_Before']

df.to_csv(f"{output_folder}/structural_metrics.csv", index=False)
```

Code snippet for structural metrics computation

	Hash	Message	MI_Before	MI_After	MI_Change	CC_Before	CC_After	CC_Change	LOC_Before	LOC_After	LOC_Change
0	014a277a97759bbc0e6ec8fba588bc6e6de65a86	A few fixes to initial point_thickness impleme...	56.611516	56.611516	0.000000	0.000000	0.000000	0.0	104.0	103.0	-1.0
1	014a277a97759bbc0e6ec8fba588bc6e6de65a86	A few fixes to initial point_thickness impleme...	NaN	NaN	NaN	NaN	NaN	NaN	230.0	238.0	8.0
2	014a277a97759bbc0e6ec8fba588bc6e6de65a86	A few fixes to initial point_thickness impleme...	NaN	NaN	NaN	NaN	NaN	NaN	455.0	455.0	0.0
3	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	100.000000	100.000000	0.000000	0.000000	0.000000	0.0	10.0	13.0	3.0
4	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	100.000000	100.000000	0.000000	0.000000	0.000000	0.0	3.0	4.0	1.0
5	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	58.276524	58.495906	0.219382	1.714286	1.714286	0.0	122.0	120.0	-2.0
6	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	NaN	58.959139	NaN	NaN	3.100000	NaN	NaN	95.0	NaN
7	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	NaN	59.449158	NaN	NaN	2.176471	NaN	233.0	102.0	-131.0
8	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	NaN	51.775973	NaN	NaN	1.961538	NaN	235.0	171.0	-64.0
9	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	NaN	NaN	NaN	NaN	NaN	NaN	198.0	184.0	-14.0
10	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	NaN	NaN	NaN	NaN	NaN	NaN	181.0	189.0	8.0
11	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	NaN	NaN	NaN	NaN	NaN	NaN	264.0	296.0	32.0
12	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	49.664046	50.088220	0.424174	2.500000	2.900000	0.4	157.0	137.0	-20.0
13	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	NaN	NaN	NaN	NaN	NaN	NaN	845.0	NaN	NaN
14	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	NaN	NaN	NaN	NaN	NaN	NaN	463.0	461.0	-2.0
15	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	100.000000	NaN	NaN	0.000000	NaN	NaN	6.0	NaN	NaN
16	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	NaN	NaN	NaN	NaN	NaN	NaN	392.0	NaN	NaN
17	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	NaN	NaN	NaN	NaN	NaN	NaN	66.0	62.0	-4.0
18	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	NaN	NaN	NaN	NaN	NaN	NaN	1706.0	1701.0	-5.0
19	2e074afb60d13262ce1e42e83bcf0ed28d95ad82	middle of massive restructure, everything stil...	64.537035	64.742479	0.205444	2.153846	2.153846	0.0	128.0	128.0	0.0

Preview of structural_metrics.csv

1.2.4 Change magnitude: semantic vs token similarity

To understand how much the code changed between the *before* and *after* versions, I measured change magnitude using two complementary metrics:

- **Semantic similarity** – Computed using CodeBERT embeddings with cosine similarity. This captures whether the two versions of the code still mean the same thing, even if the surface-level tokens look different.
- **Token similarity** – Measured using BLEU with NLTK's tokenizer (with smoothing). This focuses on how closely the literal tokens match between the two code snippets, making it sensitive to formatting and small textual edits.

I added 2 columns for these values to the dataframe and saved the dataset to `results/change_magnitude_metrics.csv`.

The following images show the code implementation for calculating the semantic similarity and token similarity:

```python
import math
from transformers import AutoTokenizer, AutoModel
from sklearn.metrics.pairwise import cosine_similarity
import nltk

nltk.download("punkt")
nltk.download("punkt_tab")

# Load CodeBERT model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("microsoft/codebert-base")
model = AutoModel.from_pretrained("microsoft/codebert-base").to(device)
model.eval()

def safe_str(s):
    if s is None:
        return ""
    if isinstance(s, float) and math.isnan(s):
        return ""
    return str(s)

def mean_pooling(model_output, attention_mask):
    token_embeddings = model_output.last_hidden_state
    input_mask_expanded = attention_mask.unsqueeze(-1).expand(token_embeddings.size()).float()
    return torch.sum(token_embeddings * input_mask_expanded, 1) / torch.clamp(input_mask_expanded.sum(1), min=1e-9)

def compute_codebert_embedding(code):
    """Compute CodeBERT embedding for given source code."""
    if not isinstance(code, str) or code.strip() == "":
        return np.zeros((768,))  # Return zero vector for empty code

    inputs = tokenizer(code, return_tensors="pt", truncation=True, padding=True, max_length=512)
    inputs = {k: v.to(device) for k, v in inputs.items()}
    with torch.no_grad():
        outputs = model(**inputs)
    # embeddings = outputs.last_hidden_state[:, 0, :].squeeze().cpu().numpy()
    embeddings = mean_pooling(outputs, inputs['attention_mask']).squeeze().cpu().numpy()
    return embeddings

def compute_semantic_similarity(row):
    """Compute semantic similarity between before and after code using CodeBERT embeddings."""
    before_code = safe_str(row['Source Code (before)'])
    after_code = safe_str(row['Source Code (current)'])

    before_embedding = compute_codebert_embedding(before_code)
    after_embedding = compute_codebert_embedding(after_code)

    if np.linalg.norm(before_embedding) == 0 or np.linalg.norm(after_embedding) == 0:
        return np.nan  # Return NaN if either embedding is zero vector

    similarity = cosine_similarity([before_embedding], [after_embedding])[0][0]
    return similarity
```

Code snippet for semantic similarity

```python
def compute_token_similarity(row):
    """Compute token similarity between before and after code using BLEU score."""
    before_code = safe_str(row['Source Code (before)'])
    after_code = safe_str(row['Source Code (current)'])

    if not before_code or not after_code:
        return np.nan  # Return NaN if either code is empty

    before_tokens = nltk.word_tokenize(before_code)
    after_tokens = nltk.word_tokenize(after_code)

    if len(before_tokens) == 0 or len(after_tokens) == 0:
        return np.nan  # Return NaN if tokenization results in empty lists

    smooth = nltk.translate.bleu_score.SmoothingFunction()
    bleu_score = nltk.translate.bleu_score.sentence_bleu([before_tokens], after_tokens, smoothing_function=smooth.method1)
    return bleu_score
```

Code snippet for token similarity

	Message	File Name	Semantic_Similarity	Token_Similarity
0	A few fixes to initial point_thickness impleme...	constants.py	0.999283	0.986973
1	A few fixes to initial point_thickness impleme...	displayer.py	1.000000	0.952839
2	A few fixes to initial point_thickness impleme...	mobject.py	0.999760	0.998018
3	middle of massive restructure, everything stil...	__init__.py	0.997906	0.660184
4	middle of massive restructure, everything stil...	__init__.py	0.996067	0.701206
5	middle of massive restructure, everything stil...	animation.py	0.999739	0.980059
6	middle of massive restructure, everything stil...	meta_animations.py	NaN	NaN
7	middle of massive restructure, everything stil...	simple_animations.py	0.994631	0.252018
8	middle of massive restructure, everything stil...	transform.py	0.993418	0.636833
9	middle of massive restructure, everything stil...	displayer.py	0.999916	0.894411
10	middle of massive restructure, everything stil...	extract_scene.py	0.999588	0.918873
11	middle of massive restructure, everything stil...	helpers.py	1.000000	0.894109
12	middle of massive restructure, everything stil...	image_mobject.py	0.999869	0.868611
13	middle of massive restructure, everything stil...	images2gif.py	NaN	NaN
14	middle of massive restructure, everything stil...	mobject.py	0.999880	0.996650
15	middle of massive restructure, everything stil...	__init__.py	NaN	NaN
16	middle of massive restructure, everything stil...	complex_multiplication_article.py	0.997389	0.655118
17	middle of massive restructure, everything stil...	generate_logo.py	0.999659	0.943530
18	middle of massive restructure, everything stil...	moser_main.py	1.000000	0.995177
19	middle of massive restructure, everything stil...	region.py	0.999849	0.989585

Table preview showing Semantic_Similarity and Token_Similarity

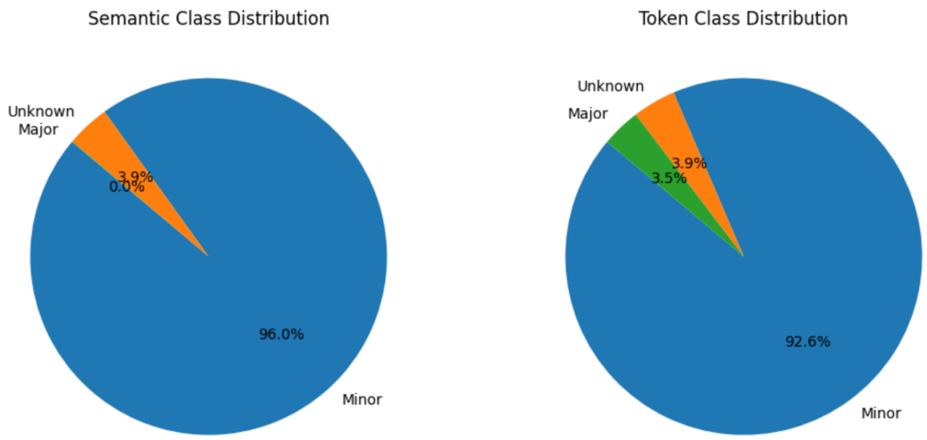
1.2.5 Classification and agreement

After computing the similarity scores, I mapped each bug-fix commit into categories of *Major* or *Minor* using simple threshold rules. This step helped in comparing how the two metrics align in their judgment of the same change.

- Semantic similarity $\geq 0.80 \Rightarrow$ *Minor*, else *Major*
- Token similarity $\geq 0.75 \Rightarrow$ *Minor*, else *Major*
- Unknown: if the metric could not be computed (NaN), the classification was recorded as *Unknown*.

```python
df = pd.read_csv(f"{output_folder}/change_magnitude_metrics.csv")

# taken from lab pdf
SEMANTIC_THRESHOLD = 0.80
TOKEN_THRESHOLD = 0.75

def classify_change(val, threshold):
    # Semantic class
    if not (isinstance(val, float) or isinstance(val, int)) or np.isnan(val):
        return 'Unknown'
    elif val >= threshold:
        return 'Minor'
    else:
        return 'Major'

df['Semantic_Class'] = df['Semantic_Similarity'].apply(lambda x: classify_change(x, SEMANTIC_THRESHOLD))
df['Token_Class'] = df['Token_Similarity'].apply(lambda x: classify_change(x, TOKEN_THRESHOLD))

df.to_csv(f"{output_folder}/final_metrics.csv", index=False)
```

Code snippet for classification



Class Distribution of Semantic_Class and Token_Class

Then, I compared the two classifications:

- If both matched, Classes_Agree = YES
- If they differed, Classes_Agree = NO
- If either was Unknown, then agreement was also Unknown

```python
def check_agreement(row):
    if row['Semantic_Class'] == 'Unknown' or row['Token_Class'] == 'Unknown':
        return 'Unknown'
    return 'YES' if row['Semantic_Class'] == row['Token_Class'] else 'NO'
df['Classes_Agree'] = df.apply(check_agreement, axis=1)

df.to_csv(f"{output_folder}/final_metrics.csv", index=False)
```

Code snippet for agreement check

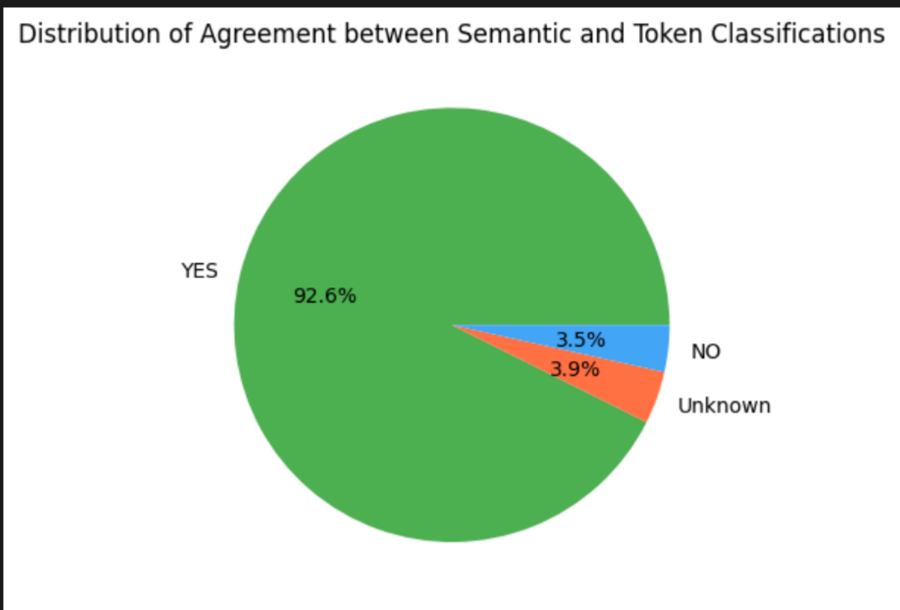
```python
df = pd.read_csv(f"{output_folder}/final_metrics.csv")
display(df[['Message', 'File Name',
            'Semantic_Class', 'Token_Class', 'Classes_Agree']
          ].head(20))
```

	Message	File Name	Semantic_Class	Token_Class	Classes_Agree
0	A few fixes to initial point_thickness impleme...	constants.py	Minor	Minor	YES
1	A few fixes to initial point_thickness impleme...	displayer.py	Minor	Minor	YES
2	A few fixes to initial point_thickness impleme...	mobject.py	Minor	Minor	YES
3	middle of massive restructure, everything stil...	__init__.py	Minor	Major	NO
4	middle of massive restructure, everything stil...	__init__.py	Minor	Major	NO
5	middle of massive restructure, everything stil...	animation.py	Minor	Minor	YES
6	middle of massive restructure, everything stil...	meta_animations.py	Unknown	Unknown	Unknown
7	middle of massive restructure, everything stil...	simple_animations.py	Minor	Major	NO
8	middle of massive restructure, everything stil...	transform.py	Minor	Major	NO
9	middle of massive restructure, everything stil...	displayer.py	Minor	Minor	YES
10	middle of massive restructure, everything stil...	extract_scene.py	Minor	Minor	YES
11	middle of massive restructure, everything stil...	helpers.py	Minor	Minor	YES
12	middle of massive restructure, everything stil...	image_mobject.py	Minor	Minor	YES
13	middle of massive restructure, everything stil...	images2gif.py	Unknown	Unknown	Unknown
14	middle of massive restructure, everything stil...	mobject.py	Minor	Minor	YES
15	middle of massive restructure, everything stil...	__init__.py	Unknown	Unknown	Unknown
16	middle of massive restructure, everything stil...	complex_multiplication_article.py	Minor	Major	NO
17	middle of massive restructure, everything stil...	generate_logo.py	Minor	Minor	YES
18	middle of massive restructure, everything stil...	moser_main.py	Minor	Minor	YES
19	middle of massive restructure, everything stil...	region.py	Minor	Minor	YES

Table showing Agreement column

I exported the final table to `results/final_metrics.csv` and plotted pie chart for the distribution of agreement column.

```
Distribution of agreement between Semantic and Token classifications:
Classes_Agree
YES        1890
Unknown      80
NO           71
Name: count, dtype: int64
```



Class Distribution of Agreement

1.3 Results and Analysis

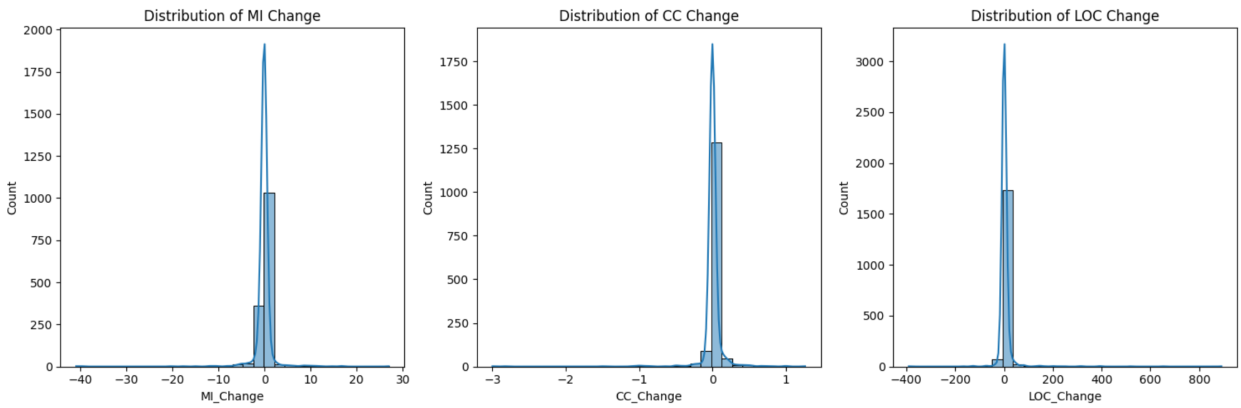
1.3.1 Final Results

Final table link: `final_metrics.csv`

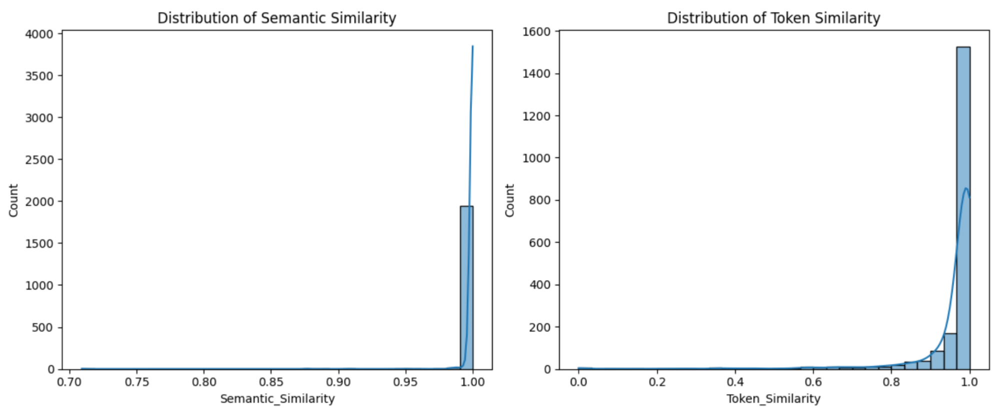
Summary of key metrics like structural changes, semantic similarity, and classification agreement:

Metric	Value
Mean MI_Change	-0.13
Mean CC_Change	0.02
Mean LOC_Change	4.5
Mean Semantic Similarity	0.9992
Mean Token Similarity	0.9596
Semantic Classification (Minor)	96.0%
Semantic Classification (Major)	0.1%
Semantic Classification (Unknown)	3.9%
Token Classification (Minor)	92.6%
Token Classification (Major)	3.5%
Token Classification (Unknown)	3.9%
Agreement (YES)	92.6%
Agreement (NO)	3.5%

1.3.2 Visualizations



Bar plots for Distribution of structural metrics



Bar plots for Distribution of Semantic and Token Similarity

1.4 Discussion and Conclusion

During this lab, I encountered a few challenges that slowed me down at first. One issue was that Radon sometimes failed when analyzing code written in older versions of Python, which meant I had to either skip those snippets or handle errors gracefully. Another challenge was that Radon itself was a completely new library for me, so I had to spend time going through its documentation and experimenting before I could use it confidently. I also ran into problems with the NLTK tokenizer setup – the lab notebook would throw runtime errors until I figured out that the punkt package needed to be downloaded separately.

This lab helped me learn a lot. I now have a much better understanding of **structural metrics** like Maintainability Index (MI), Cyclomatic Complexity (CC), and Lines of Code (LOC), and how they can reflect code quality changes. On the other hand, exploring **semantic similarity with CodeBERT** and **token similarity with BLEU** showed me how different perspectives can highlight different aspects of the same bug fix.

Overall, this lab felt like a natural extension of Lab 2. It pushed me to look beyond raw diffs and I learnt how we can classify a change/bugfix as “major” or “minor” by combining structural and semantic metrics.

1.5 References

[1] Radon documentation

[2] CodeBERT model (Hugging Face)

[3] NLTK tokenizer

[4] Lab Document (Google Doc)