ICSME 2023 | 1 – 6 October 2023

GPTCloneBench: A comprehensive benchmark of semantic clones and cross-language clones using GPT-3 model and SemanticCloneBench

By: Ajmain I. Alam, **Palash R. Roy**, Farouq Al-omari, Chanchal K. Roy, Banani Roy, Kevin A. Schneider

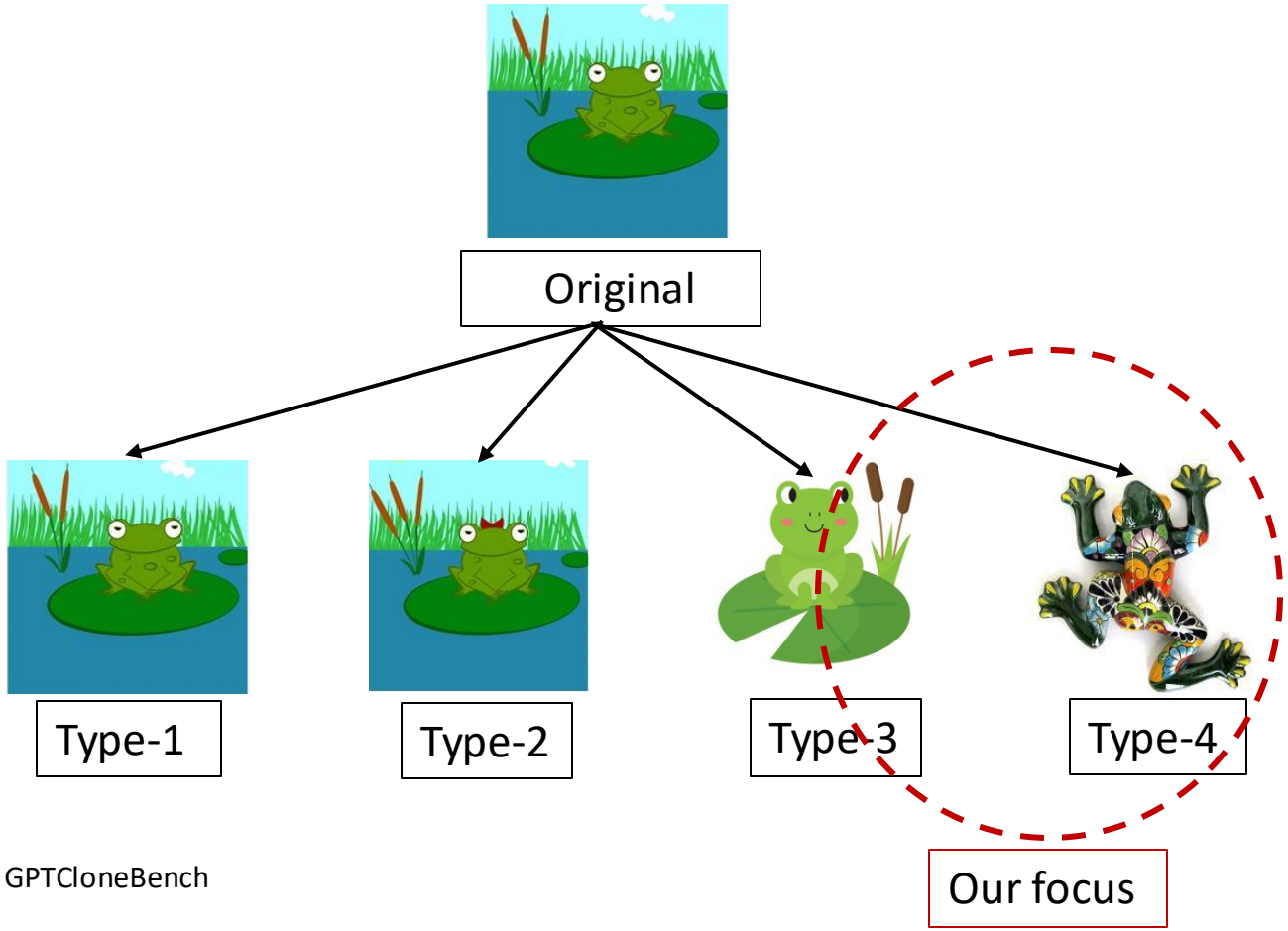
Software Research Lab

[SOAR]

iSE LAB
Interactive Software Engineering and Analytics Lab

Awarded with **Open Research Object** Badge

Definition



Semantic Clones

```python
def factorial(n):
    x = n
    for i in range(1,n,1):
        x = x*(n-i)
    return x
```

```python
def factorial(n):
    if n <= 1:
        return 1
    else:
        return n * factorial(n-1)
```

Cross-language Clones

```java
// Java
public static boolean longerTF (boolean [] guess) {
    int variableTrue = 0;
    int variableFalse = 0;
    for (int x = 0;
    x < guess.length; x ++) {
        if (guess [x]) {
            variableTrue ++;
        } else {
            variableFalse ++;
        }
    }
    return variableTrue >= variableFalse;
}
```

```python
// Python
def longerTF(guess):
    variableTrue = 0
    variableFalse = 0
    for x in range(len(guess)):
        if (guess[x]):
            variableTrue += 1
        else:
            variableFalse += 1
    return variableTrue >= variableFalse
```

Background



WHAT

- Copy/Paste
- Code reuse
- Similar problem-solving approach



WHY

- Saves Time
- Meeting deadlines
- Following established patterns



CHALLENGES

- Maintenance
- Bug fixing
- Code quality

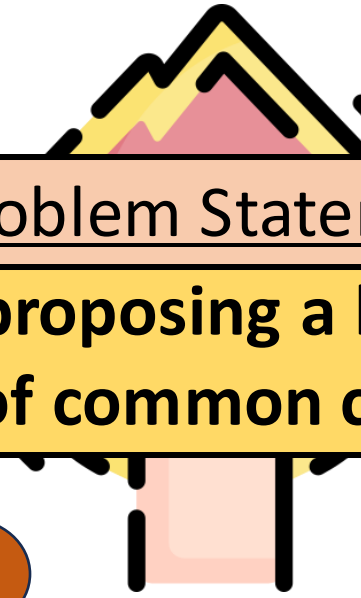
Motivation

01

Semantic clones are being the most complex and challenging to detect.

03

SemanticCloneBench has less number of clones to train and test a Machine Learning model.



Problem Statement

In this study, we are proposing a benchmark to overcome the limitations of common clone benchmarks.

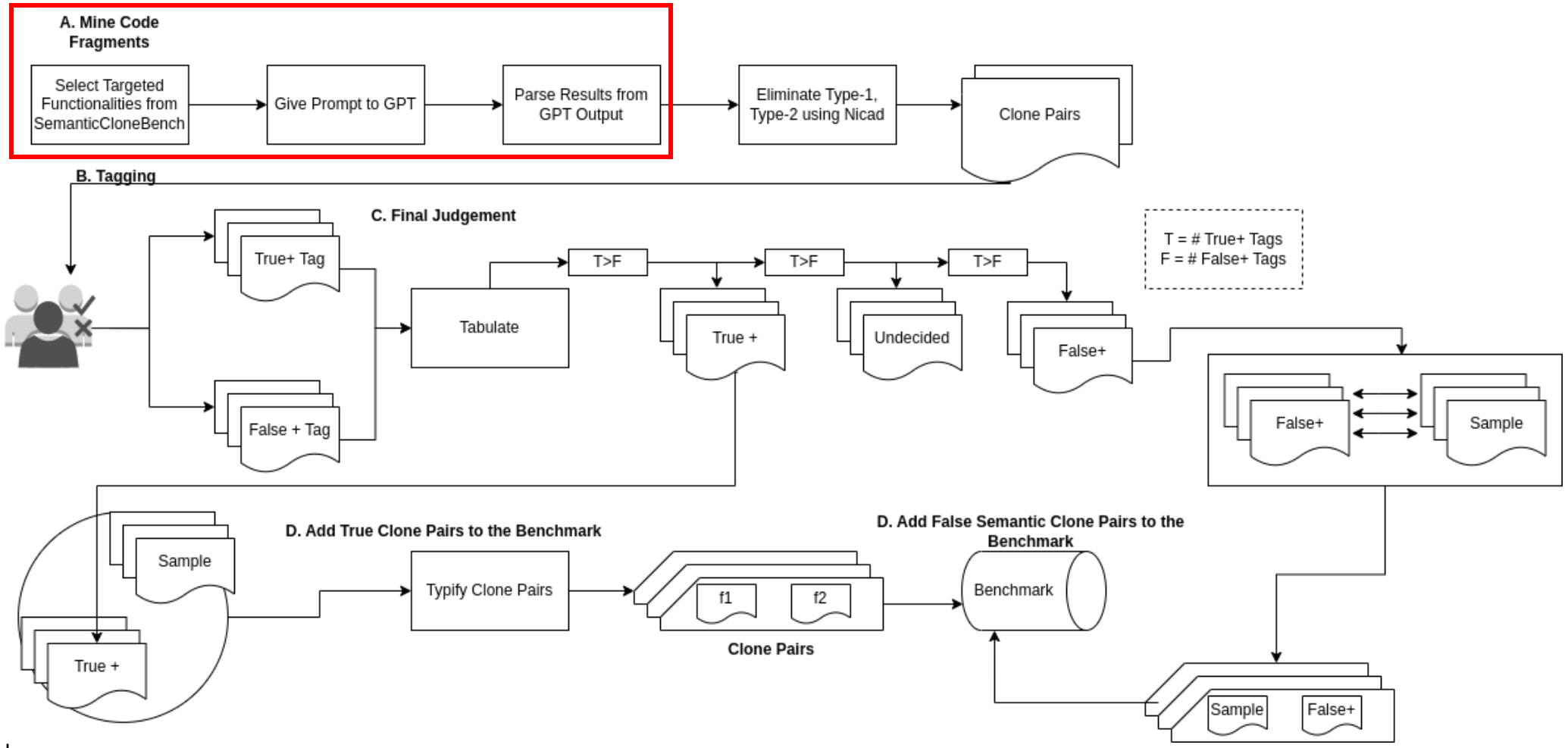
02

BigCloneBench, originally is not designed for semantic clone detection.

04

CLCDSA only contains cross-language clones of different programming contest solutions.

Building GPTCloneBench (Overall methodology)



Building GPTCloneBench (Fetching from GPT-3)

- Select Target Functionality from SemanticCloneBench
- *Few-shot prompt engineering*
- *Developed automated script*
- 4 programming languages



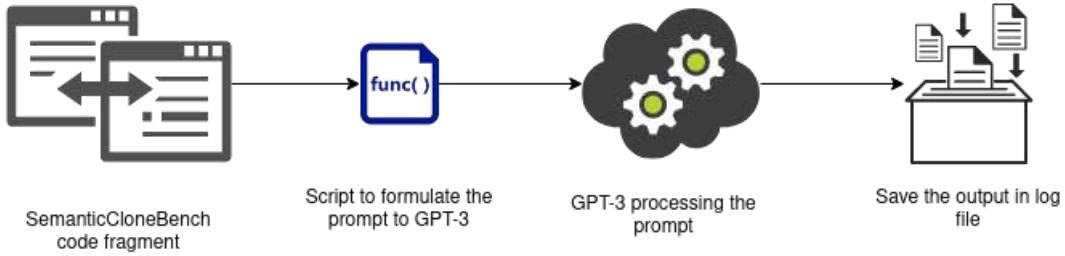


Figure: GPT-3 output fetching process

Few-shot prompts

- Give me 4 **Type-3, Type-4 clone** for the following code *<code fragment>*
- Give me 10 **distinctive** implementation for the following code *<code fragment>*
- Give me **different** implementation for the following code *<code fragment>*
- Give me **unique** implementation for the following code *<code fragment>*
- Give me different **functional** implementation for the following code *<code fragment>*
- Give me different **syntax** implementation for the following code *<code fragment>*

Results of different prompt

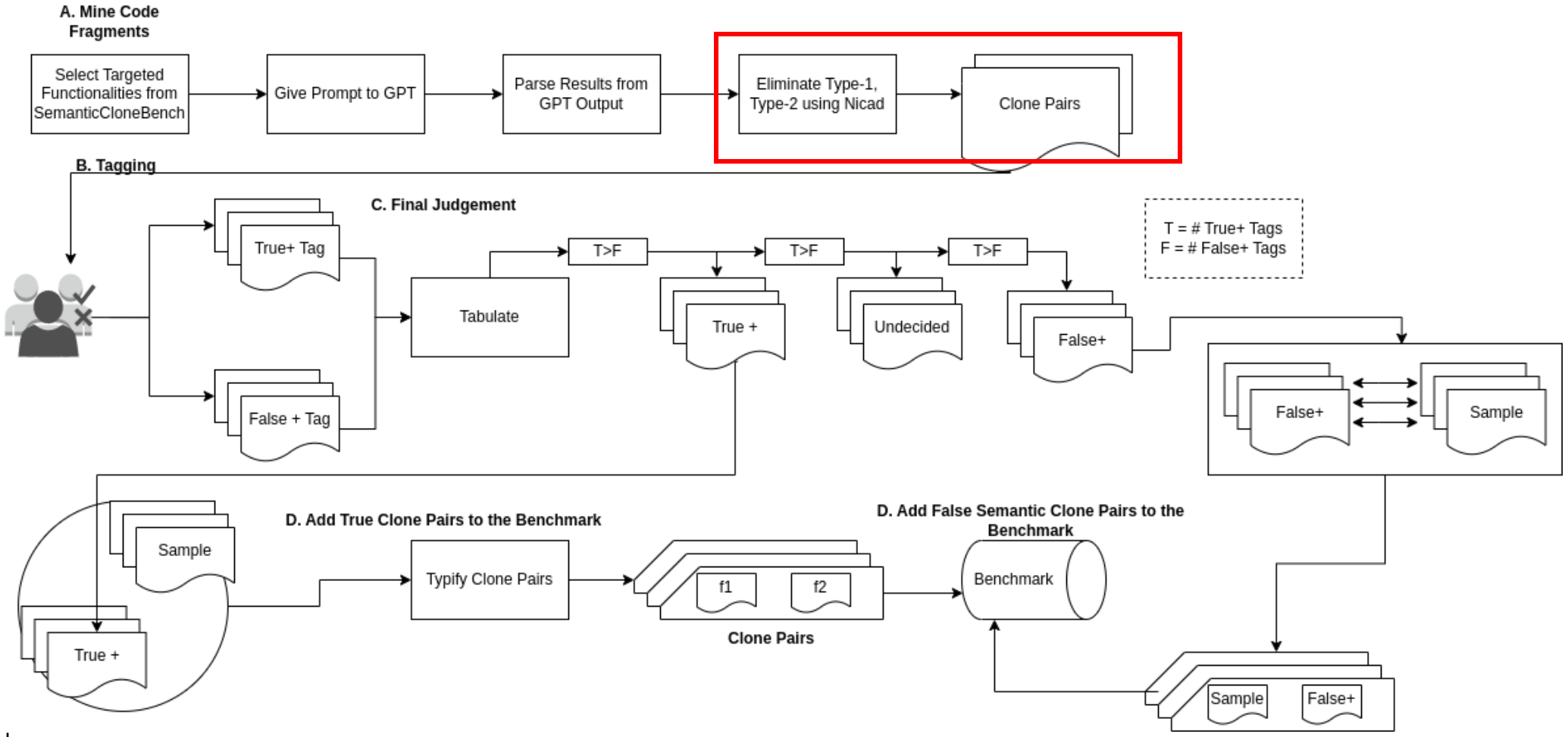
SUMMARY RESULTS OF SIMILARITY % FOR EACH QUERY

Language	Type-3/4	Different	Unique	Functional	Syntax	Distincitive
Python	74.75	40.4	36.5	27.1	43.1	31.9
Java	100	34.55	34.55	59.5	66.2	34.55
C	100	74.9	82.2	41.6	91	46.5
C#	100	67.33	63.33	81.2	100	80.5

Finalized prompts

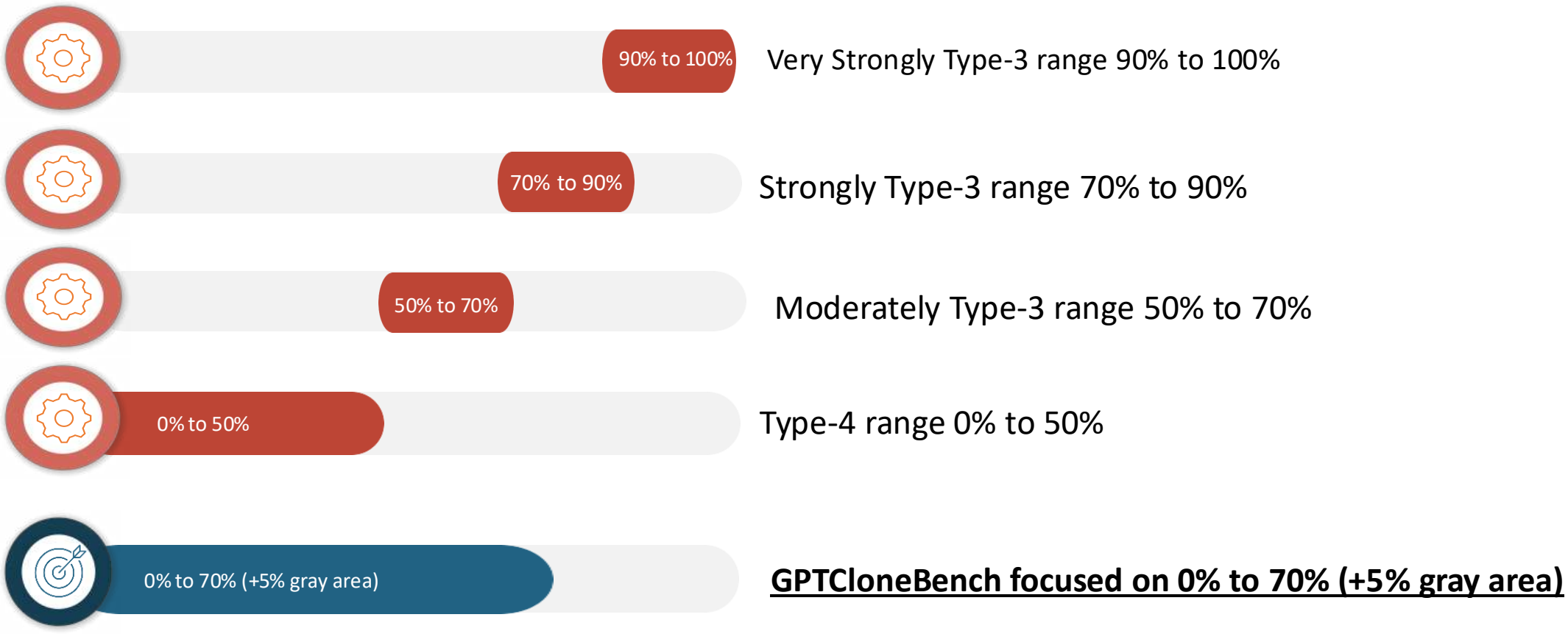
- Prompt 1: Give me **Type-3, Type-4 clone** for the following code *<code fragment>*
- Prompt 2: Give me 10 **distinctive** implementation for the following code *<code fragment>*

Building GPTCloneBench (Overall methodology)



Building GPTCloneBench (Syntactic Clones Filtration)

Followed the categorization scheme of BigCloneBench/BigCloneEval for code similarity

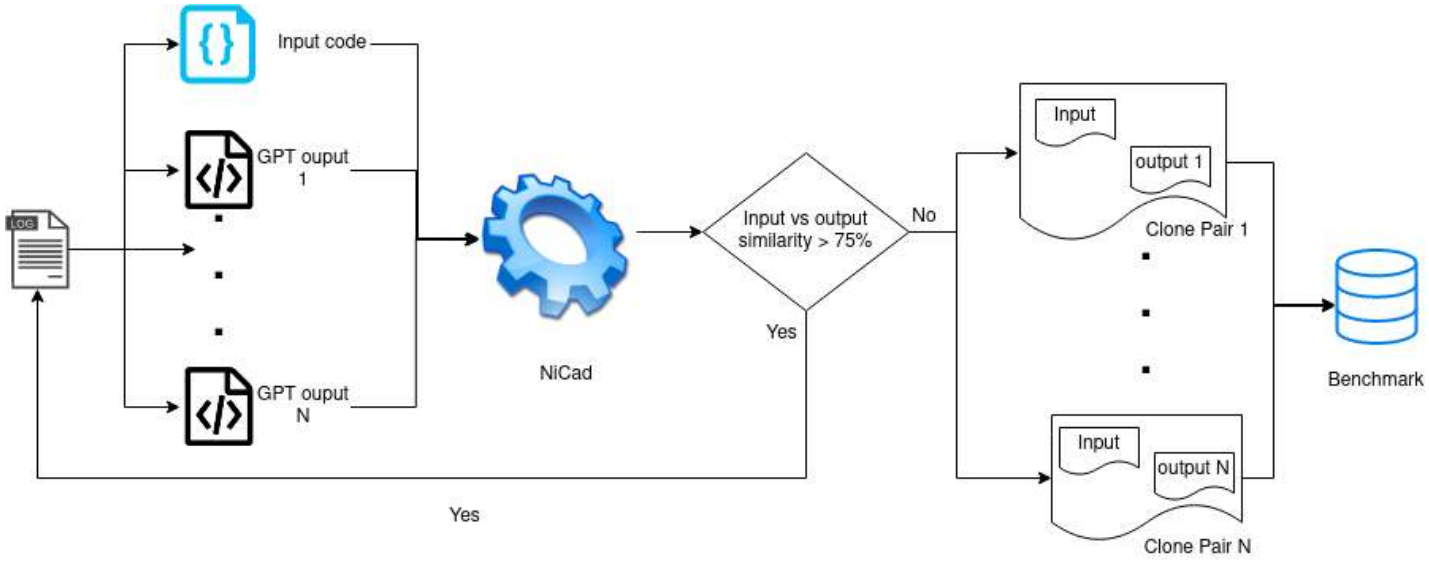


BigCloneEval source: https://ieeexplore.ieee.org/document/7816515

Building GPTCloneBench (Syntactic Clones Filtration)

NiCad Configuration:

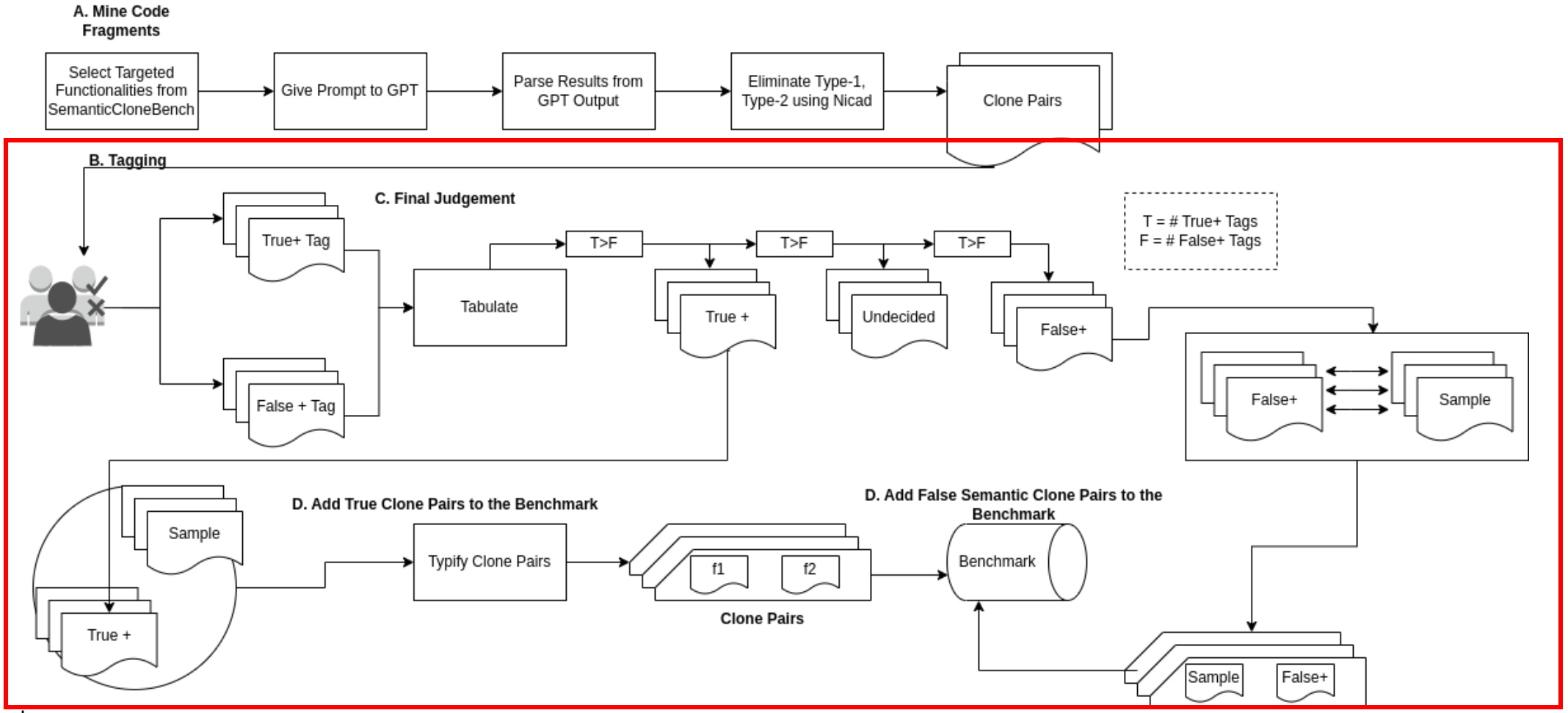
- Threshold: 99%
- Min Line: 03
- Renaming: Blind



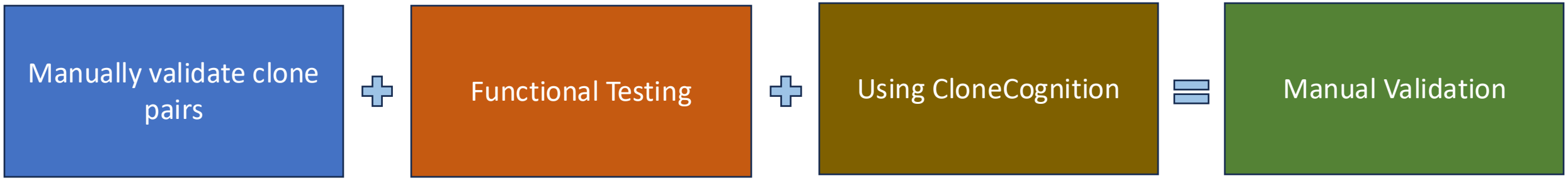
Filtered out clones: if NiCad similarity > 75%

NiCad source: https://txl.ca/txl-nicaddownload.html

Building GPTCloneBench (Overall methodology)



Building GPTCloneBench (Manual Validation)



Building GPTCloneBench (Manual Validation Continued)

- **Manually validate clone pairs:**
  - Nine Judges (6 Undergrad research students, 3 post-doctoral researchers)
  - Total Pairs after filtering out syntactic clones: 38,276 Semantic Pairs, 22,364 Cross language pairs.
  - Three groups
  - Discussion

- **Functional Testing: Randomly tested 1000 clone pairs.**

- **True Clone Pairs:**
  - Semantic clones: 37,149
  - Cross-language clones: 20,770
  - False Semantic (Syntactic clones): 19,288

Clone Pair	Judge 1	Judge 2	Decision
Pair 1	True	True	Selected
Pair 2	False	False	Discarded
Pair 3	Undecided	Undecided	Going to next step
Pair 4	True	False	Discarded
Pair 5	False	Undecided	Discarded

Table 1: Judges decision making rules

Tools used for manually validate clone pairs:
https://github.com/jeffsvajlenko/ValidateClones

Validation With CloneCognition

- Used this tool to classify clones in the benchmark.
- Used it in reverse way.
- The lower the accuracy, the better the result of the benchmark.

CloneCognition Tool:
https://github.com/pseudoPixels/CloneCognition

CloneCognition
paper: https://clones.usask.ca/pubfiles/articles/MostaeenCloneCognitionFSE2019.pdf

CloneCognition Results with Type-3, Type-4 Prompt

Language	Threshold	# Clones	Predicted	Accuracy
Python	76%	1495	229	0.15
Java	76%	1087	397	0.37
C	76%	563	206	0.37
C#	76%	1033	330	0.32

CloneCognition Results with Distinctive Prompt

Language	Threshold	# Clones	Predicted	Accuracy
Python	76%	4968	228	0.05
Java	76%	4635	516	0.11
C	76%	2853	511	0.18
C#	76%	4025	394	0.098

GPTCloneBench Details

Benchmark is divided into two types of system

- Stand-alone clones (A clone pair is saved in a single file).
  - True Semantic Clones
    - Divided based on language. Each language has separate folder.
  - False Semantic Clones
- System-Injected clones (Clones injected in the system). Injected system list:
  - JHotDraw (for Java)
  - PostgreSQL (for C)
  - Mono (for C#)
  - Django (for Python)

Benchmark can be found in these two links:

- GitHub: https://github.com/srlabUsask/GptCloneBench
- Zenodo: https://zenodo.org/record/8259988
- Paper link: https://ieeexplore.ieee.org/document/10336319

GPTCloneBench in Use

- Used three clone detection tools.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Using stand-alone clones

SourcererCC, Oreo and CLCDSA Recall results

Tool	Language	Granularity	Recall
SourcererCC	Java	Method	0.135
	C	Method	0.12
	C#	Method	0.10
Oreo	Java	Method	0.49
CLCDSA	Java and C#	Method	0.19
	C# and Python	Method	0
	Java and Python	Method	0.001

Using system-injected clones

SourcererCC and CLCDSA results with injected systems

Tool	Language	Granularity	Precision	Recall
SourcererCC	Java	Method	0.82	0.33
CLCDSA	Java, C#, Python	Method	0.47	0.10

Threats to validity

- Real vs artificial clones.
- Biasness in manual validation.
- Prompt Engineering.



Challenges

While working with GPT-3, we have faced following challenges:

- Token Limitation of OpenAI API
- GPT-3 output does not follow a fixed pattern. Data processing was difficult. Automated script generation and some manual inspection were required.
- Cost for GPT-3 API usage



Conclusion

- This research aims to propose a novel approach for building a comprehensive benchmark of semantic and cross-language clones using the GPT-3 model.

- This initial edition of the GPTCloneBench includes over 37,149 true semantic clones, 20,770 cross-language pairs and 19,288 false semantic (Type-1/Type-2) clone pairs.

- Can be used for other purposes:
  - Comparing and evaluating semantic and cross-language clone detection tools.
  - Evaluate whether source code transformation-based clone detection tools such as CloneWorks or SimCad could in fact detect semantic clones by applying flexible source transformations and normalizations.
  - Build IDE-based flexible clone detection and management.
  - This approach can potentially be used in building similar benchmarks in other contexts.

- In future we will add more data to our benchmark.

- **Paper link: https://ieeexplore.ieee.org/document/10336319**

Thank you

LinkedIN



Acknowledgment

This work was supported in-part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery grants, the John R. Evans Leaders Fund (JELF) of the Canada Foundation for Innovation (CFI), and NSERC CREATE graduate program on Software Analytics Research (SOAR) grants.