Code Clone Detection Using Semantic and Syntactic Properties

BLG 630 – RECOMMENDATION SYSTEMS IN SOFTWARE ENGINEERING

HALİT UYANIK

504202506

Table of contents

Introduction

- Code clones can increase maintenance cost in the long run.
  - Making changes takes more time
  - Introducing bugs propagates across clones.
- Clones can be detected using graph, token, design elements.
- Goals:
  - How to find and extract syntactic and semantic features of a code?
  - How to train a model using these features for detecting clones?
  - What is the difference between detecting detecting different clone types?
  - How to implement a simple UI for a clone recommendation engine?

Related Work

- Semantic Clone Detection Using Machine Learning
  - Extracting syntactic and semantic features
  - Evaluated using BigCloneBench and IJaDataset
  - Comparison with existing approaches

- Improvements:
  - Only 10-Fold cross validation results are reported, equal train-test split
  - Only a small portion of the used features are given
  - No tool implementation goal

Clone Types

- Type1
- Type2
- Type3/4
  - Type3
  - Type4
  - VST3 [0.9, 1.0]
  - ST3 [0.7, 0.9)
  - MT3 [0.5, 0.7)
  - WT3 (0.0, 0.5)

Tools

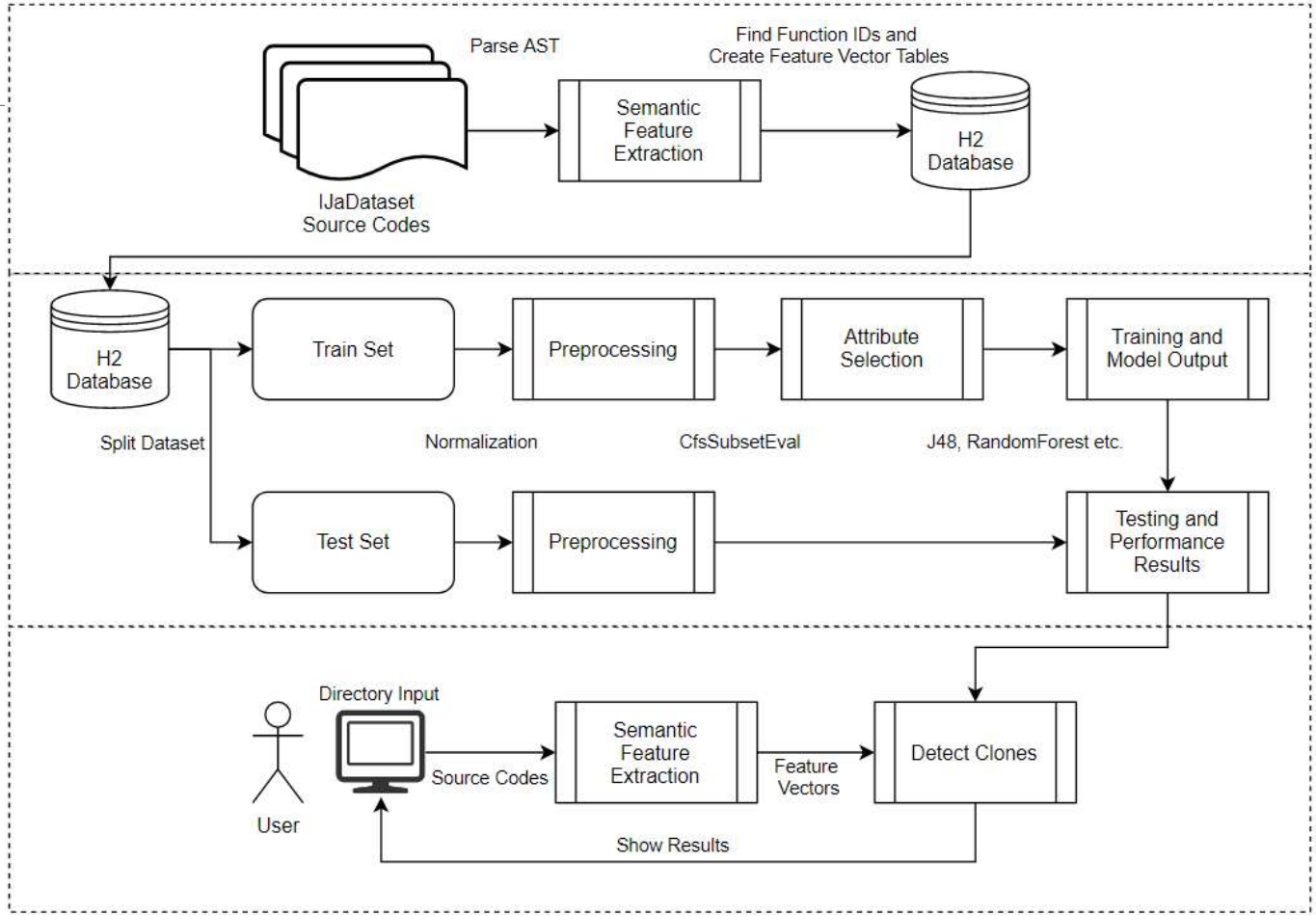
- JavaParser
  - AST and PDG analysis
- Weka
  - Preprocessing and machine learning
  - Wide documentation and community support
- H2Database
  - BigCloneBench dataset
  - Data processing and ground truths

BigCloneBench Dataset

- Public dataset
  - Continously updated
- Includes two main resources
  - Clone dataset
  - IJaDataset source codes
- 8 million clone pairs
- 3% False Positives
- 95% WT3 clones
- 760 000 methods in IJaDataset
- Only 8 million of 288 million possible pairs are clones

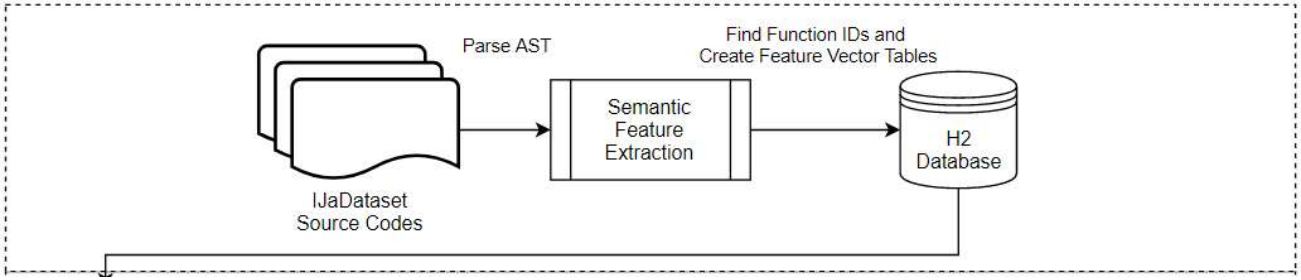
Clone Type	Count	Ratio
Type1	48116	0.5%
Type2	4234	0.04%
VST3	4577	0.05%
ST3	16818	0.1%
MT3	86341	0.9%
WT3	8424067	95%
False Positives	279032	3%
Total	8863185	100%

Recommendation System Design



Feature Extraction

- Java program is written
- Extract every single method body from IJaDataset source code files
- Convert them into AST format as node representation and extract 86 features
- Example features:
  - If-else counts, for-foreach-while-do loops etc.
  - Number of assignments after conditional checks etc.

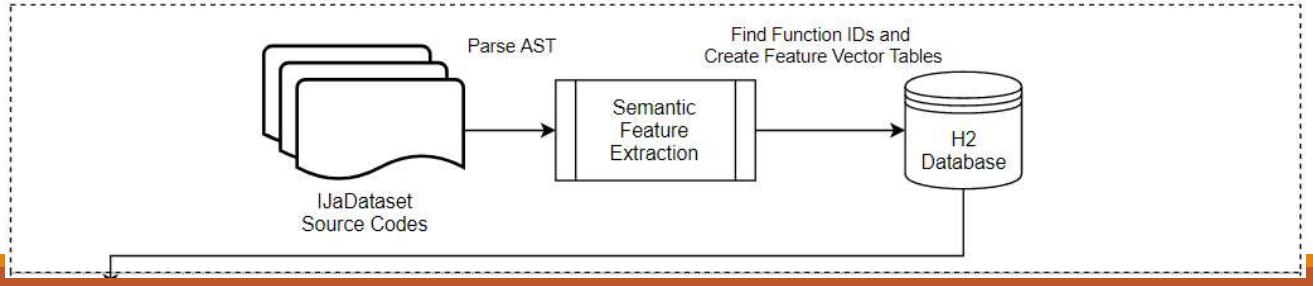


Feature Extraction

- Save extracted features and their classname-paths in H2Database
- Find and match function IDs, now clone ground truths can be matched as well
- Feature vectors are created as method feature pairs with their ground truth.

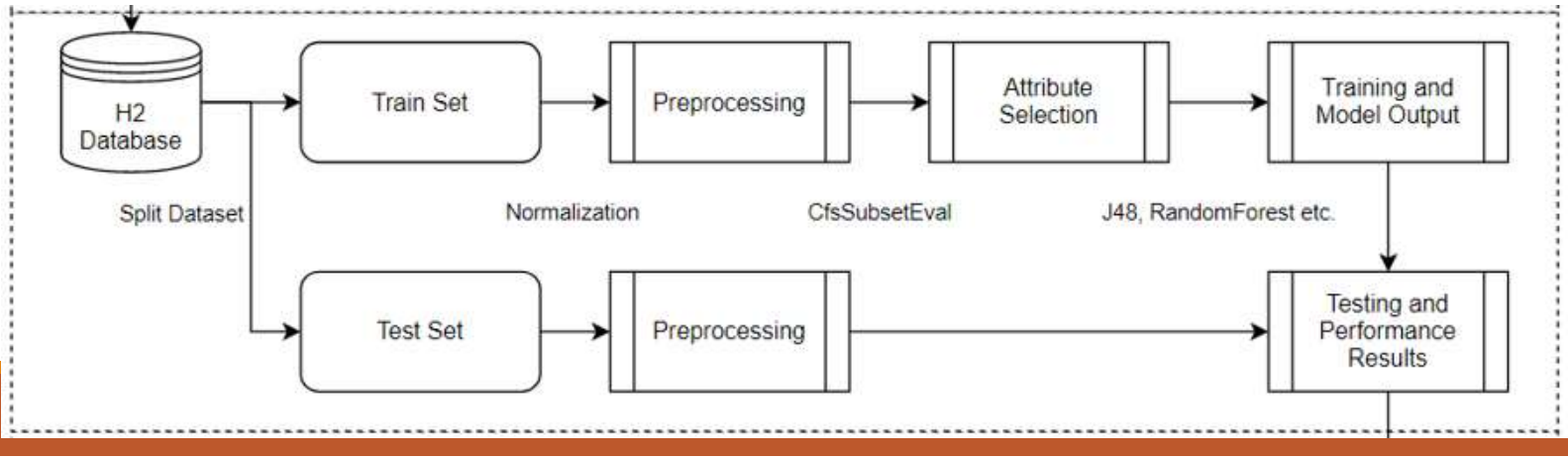
FV = [X1, X2, …. X86, Y1, Y2, … Y86, Difference, Clone]

- Alternative representations exists which improves performance and storage but reduces performance.



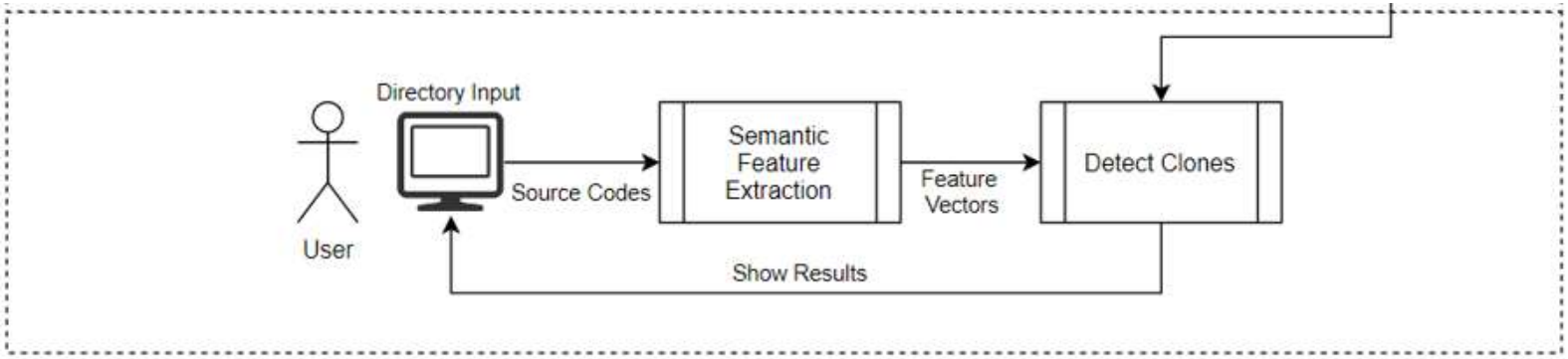
Model Training

- Fetch the feature vectors, nearly 1.8 million vector is fetched.
- Randomize the dataset and split train-test set
- Normalization and Attribute selection on the train set
  - These are also applied to test set during testing phase
- Train the model using train split
- Test and report results using the test split



Using Saved Model

- Best model obtained in the previous step is compiled into a .jar file
- Trained model itself is also saved
- User inputs a directory from a Winform UI
- Feature vectors are extracted and labelled using the compiled .jar code
- Results are shown in the interface
- More information will be given further in the presentation



Experiments

- Two different feature types
  - 10 widely used basic features
  - 86 features in this study
- Two different evaluation approaches
  - 10-Fold Cross Validation (used in the referenced paper as well)
  - Train-Test split
- Two different Train-Test split distributions
  - Equal number of clones
  - Real ratio between number of clones according to BigCloneBench database
- Three different models: J48, RandomForest, RandomCommittee
- Recall, Precision, F-Measures, and Confusion Matrices are reported

Train-Test Distributions

Real Ratio Distribution			
Clone Type	Train Count	Test Count	Total
Type1	3360	38184	41544
Type2	3360	840	4200
VST3	3360	908	4268
ST3	3360	3337	6697
MT3	3360	17130	20490
WT3	3360	1671284	1674644
False Positive	3360	55358	58718
SUM	23520	1787041	1810561

Equal Ratio Distribution			
Clone Type	Train Count	Test Count	Total
Type1	3360	840	4200
Type2	3360	840	4200
VST3	3360	840	4200
ST3	3360	840	4200
MT3	3360	840	4200
WT3	3360	840	4200
False Positive	3360	840	4200
SUM	23520	5880	29400

10-Fold Cross Validation Results

- Shows expected results for equal scenario.
- Using all features is better than standart 10 features.
- RandomForest performs best however performance falls for Type3 clones.

Algorithm	Features	Clone Type	Precision (%)	Recall (%)	F Measure (%)
J48	Default Ten Features	TYPE1	90,7	93,2	91,9
		TYPE2	82,7	88,8	85,6
		VST3	76,4	81	78,6
		ST3	74,3	68,6	71,3
		MT3	73,5	72,4	72,9
		WT3	84,9	78,4	81,5
		FP	94,5	95,4	95
		Overall	82,4	82,5	82,4
	All Features	TYPE1	98,8	97,9	98,3
		TYPE2	96,9	98,1	97,5
		VST3	93,6	93,6	93,6
		ST3	84,1	86,8	85,4
		MT3	79,6	80,1	79,9
		WT3	89,3	85,3	87,2
		FP	97,1	97,3	97,2
		Overall	91,3	91,3	91,3

Algorithm	Features	Clone Type	Precision (%)	Recall (%)	F Measure (%)
Random Forest	Default Ten Features	TYPE1	92,9	93,3	93,1
		TYPE2	85	88,8	86,9
		VST3	81,2	82,1	81,6
		ST3	80,8	71	75,6
		MT3	78,7	82,6	80,6
		WT3	89,5	89,5	89,5
		FP	96,4	97,7	97
		Overall	86,4	86,4	86,3
	All Features	TYPE1	99,9	97,4	98,6
		TYPE2	96,7	95,7	96,2
		VST3	94,2	91,7	92,9
		ST3	91,3	87,7	89,5
		MT3	82	90,4	86
		WT3	90,6	90,9	90,7
		FP	98,9	98,6	98,8
		Overall	93,4	93,2	93,2

Algorithm	Features	Clone Type	Precision (%)	Recall (%)	F Measure (%)
Random Committee	Default Ten Features	TYPE1	92,1	93,4	92,7
		TYPE2	83,9	89	86,4
		VST3	79,6	82	80,8
		ST3	79,1	71,3	75
		MT3	77,7	79,9	78,8
		WT3	89,9	87	88,4
		FP	97,1	96,9	97
		Overall	85,6	85,6	85,6
	All Features	TYPE1	99,6	97,5	98,5
		TYPE2	95,9	96,2	96
		VST3	93,4	92,1	92,7
		ST3	89	88,5	88,7
		MT3	81,8	87,9	84,7
		WT3	91,1	88,6	89,8
		FP	98,8	98	98,4
		Overall	92,8	92,7	92,7

Equal Train-Test Split Results

- Again, using all features is better
- RandomCommittee is better than RandomForest for WT3 and MT3 clones.
- Similar results to cross validation as expected.

Algorithm	Features	Clone Type	Precision (%)	Recall (%)	F Measure (%)
J48	Default Ten Features	TYPE1	90,6	92,7	91,6
		TYPE2	81,3	89,3	85,1
		VST3	75,8	80,1	77,9
		ST3	76,2	68,6	72,2
		MT3	73,7	74,4	74,1
		WT3	87,5	78,9	83
		FP	95,3	96,2	95,7
		Overall	82,9	82,9	82,8
	All Features	TYPE1	98,9	96,8	97,8
		TYPE2	96,6	97,6	97,1
		VST3	94,7	94	94,4
		ST3	85,8	86,2	86
		MT3	78,3	81,9	80
		WT3	89,2	86	87,6
		FP	97,6	98	97,8
		Overall	91,6	91,5	91,5

Algorithm	Features	Clone Type	Precision (%)	Recall (%)	F Measure (%)
Random Forest	Default Ten Features	TYPE1	91,9	93	92,4
		TYPE2	84,8	89,9	87,3
		VST3	81,7	81,3	81,5
		ST3	83,1	69,6	75,8
		MT3	80,1	86,5	83,2
		WT3	91,8	90,1	90,9
		FP	95,2	98,2	96,7
		Overall	86,9	87	86,8
	All Features	TYPE1	99,9	96,5	98,2
		TYPE2	96,7	95,1	95,9
		VST3	94,5	91,5	93
		ST3	91,2	86,7	88,9
		MT3	80,7	91	85,5
		WT3	90,7	91,5	91,1
		FP	99,2	98,6	98,9
		Overall	93,3	93	93,1

Algorithm	Features	Clone Type	Precision (%)	Recall (%)	F Measure (%)
Random Committee	Default Ten Features	TYPE1	92	93	92,5
		TYPE2	83,5	89,9	86,6
		VST3	79,4	81,7	80,5
		ST3	79,6	70,6	74,8
		MT3	79,2	82,9	81
		WT3	91,9	86,8	89,3
		FP	96,8	97,5	97,2
		Overall	86,1	86	86
	All Features	TYPE1	99,8	96,5	98,1
		TYPE2	96,1	96,4	96,3
		VST3	94	91,9	93
		ST3	89	87,7	88,4
		MT3	81,7	89,3	85,3
		WT3	91,1	89,4	90,3
		FP	98,8	98	98,4
		Overall	92,9	92,8	92,8

Real Ratio Train-Test Split Results

- Again, using all features is better.
- Would be interesting to argue for attribute selection since all features might be important.
- RandomForest performs best, however Type2 and Type3 clone precisions are quite low.

Algorithm	Features	Clone Type	Precision (%)	Recall (%)	F Measure (%)
J48	Default Ten Features	TYPE1	77,7	93	84,7
		TYPE2	6,1	89,3	11,3
		VST3	2	83,9	3,9
		ST3	3,9	69,2	7,4
		MT3	6,5	73,9	11,9
		WT3	99,8	79,8	88,7
		FP	52,1	95,5	67,5
		Overall	96,7	80,5	87
	All Features	TYPE1	100	97,4	98,7
		TYPE2	52,8	98,2	68,7
		VST3	22,6	93,5	36,4
		ST3	9,9	86,8	17,7
		MT3	7	79,8	12,9
		WT3	99,9	86,3	92,6
		FP	70	97,4	81,4
		Overall	97,8	86,8	91,4

Algorithm	Features	Clone Type	Precision (%)	Recall (%)	F Measure (%)
Random Forest	Default Ten Features	TYPE1	93	93	93
		TYPE2	35,6	89,8	51
		VST3	11,6	84,3	20,3
		ST3	22,5	70,5	34,1
		MT3	10,2	85,9	18,3
		WT3	99,9	89,1	94,2
		FP	56,3	97,6	71,4
		Overall	97,3	89,3	92,5
	All Features	TYPE1	100	97	98,5
		TYPE2	68,2	95,5	79,5
		VST3	40,4	90,5	55,9
		ST3	37,9	86,2	52,7
		MT3	10,6	90,4	19
		WT3	99,9	91,3	95,4
		FP	82,6	98,6	89,9
		Overall	98,3	91,6	94,5

Algorithm	Features	Clone Type	Precision (%)	Recall (%)	F Measure (%)
Random Committee	Default Ten Features	TYPE1	90,2	93,1	91,6
		TYPE2	21,8	89,9	35,1
		VST3	6,4	83,3	11,8
		ST3	12,2	71,6	20,9
		MT3	8,7	82,5	15,8
		WT3	99,9	87,6	93,3
		FP	63,1	97,2	76,6
		Overall	97,4	87,9	91,8
	All Features	TYPE1	97,9	97,1	97,5
		TYPE2	48	96,6	64,1
		VST3	13,1	90,2	22,9
		ST3	20,9	87,1	33,8
		MT3	8,4	87,4	15,4
		WT3	99,9	88,7	94
		FP	82,9	98,1	89,9
		Overall	98,2	89,2	93

Confusion Matrix

- Confusion matrix for best real ratio performance model
- No false positives before below 50% similarity
- Type 3/4 clones are harder to distinguish
- Number of WT3 affects the results immensely
- Ratio difference between Type3 clones and borderlines

Class	TYPE1	TYPE2	VST3	ST3	MT3	WT3	nonmatch
TYPE1	37046	335	466	103	102	133	0
TYPE2	1	803	17	9	9	2	0
VST3	0	15	774	30	22	14	0
ST3	0	13	96	2878	295	56	0
MT3	0	0	53	745	15479	793	61
WT3	13	12	508	3820	129526	1525943	11463
nonmatch	0	0	0	2	95	693	54569

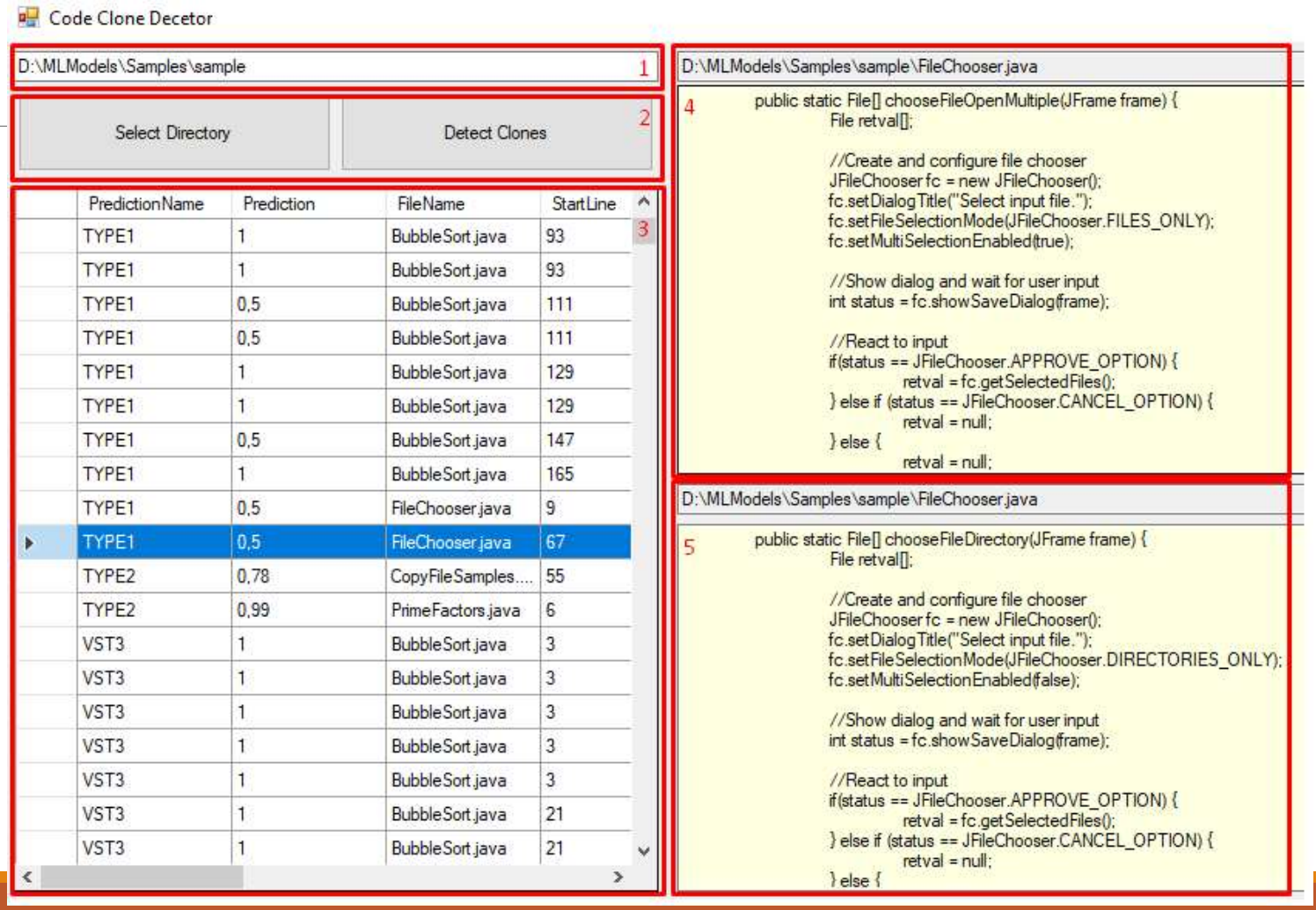
Previous Studies

- Tried testing the tools with BigCloneBench but results were not correlated with previous given ones.
- For equal distribution results model performs better than most except NiCad VST3 and ST3.
- Real ratio performs better for MT3 and WT3/4.

Tool	Type of Clone	Recall	Precision	F-Measure
SourcererCC	VST3 ST3 MT3 WT3/4	93% 61% 5% 0%	91% (as reported)	92% 73% 9% 0%
CCFinder	VST3 ST3 MT3 WT3/4	62% 15 % 1% 0%	$\approx 60\% - 72\%$ (as reported)	$\approx 61\% - 67\%$ $\approx 24\% - 25\%$ $\approx 2\%$ 0%
Deckard	VST3 ST3 MT3 WT3/4	62% 31% 12% 1%	**93% (as reported)**	74% 47% 21% 2%
iClones	VST3 ST3 MT3 WT3/4	82% 24% 0% 0%	(Unreported)	(Unreported)
NiCad	VST3 ST3 MT3 WT3/4	**100%** **95%** 1% 0%	$\approx 80\% - \mathbf{96\%}$ (as reported)	$\approx 89\% - \mathbf{98\%}$ $\approx 87\% - \mathbf{95\%}$ $\approx 2\%$ 0%

Clone Detection Tool

- 1 – Directory Input
- 2 – Buttons
- 3 – Predictions
- 4 – Source Code
- 5 – Target Code
- Tool may not detect all types properly.
  - Look at JFileChooser.FILES_ONLY
  - JFileChooser.DIRECTORIES_ONLY



Conclusion & Discussion

- A feature based model and its tool is implemented.
- How many features are enough?
- Lack of relational features.
- Token based approach might be better for more performance and less cost.
- Using different models.
- Effects of lack of expertise.

References

[1] “Java documentation - get started,” Dec 2020. [Online]. Available:https://docs.oracle.com/en/java/

[2] Adegeo,“Windowsformsfor.net5docu-mentation.”[Online].Available:https://docs.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-5.0

[3] A. Sheneamer and J. Kalita, “Semantic clone detection using machinelearning,” in2016 15th IEEE International Conference on MachineLearning and Applications (ICMLA). IEEE, 2016, pp. 1024–1028.

[4] “Bigcloneeval.”[Online].Available:https://jeffsvajlenko.weebly.com/bigcloneeval.html

[5] Javaparser,“javaparser/javaparser.” [Online]. Available: https://github.com/javaparser/javaparser

[6] “Weka.” [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/

[7] [Online]. Available: https://www.h2database.com/html/main.html