Statistical-based Clone Detection

He Feng
Department of Physics
fenghe@vt.edu

Liuqing Li
Department of Computer Science
liuqing@vt.edu

Outline

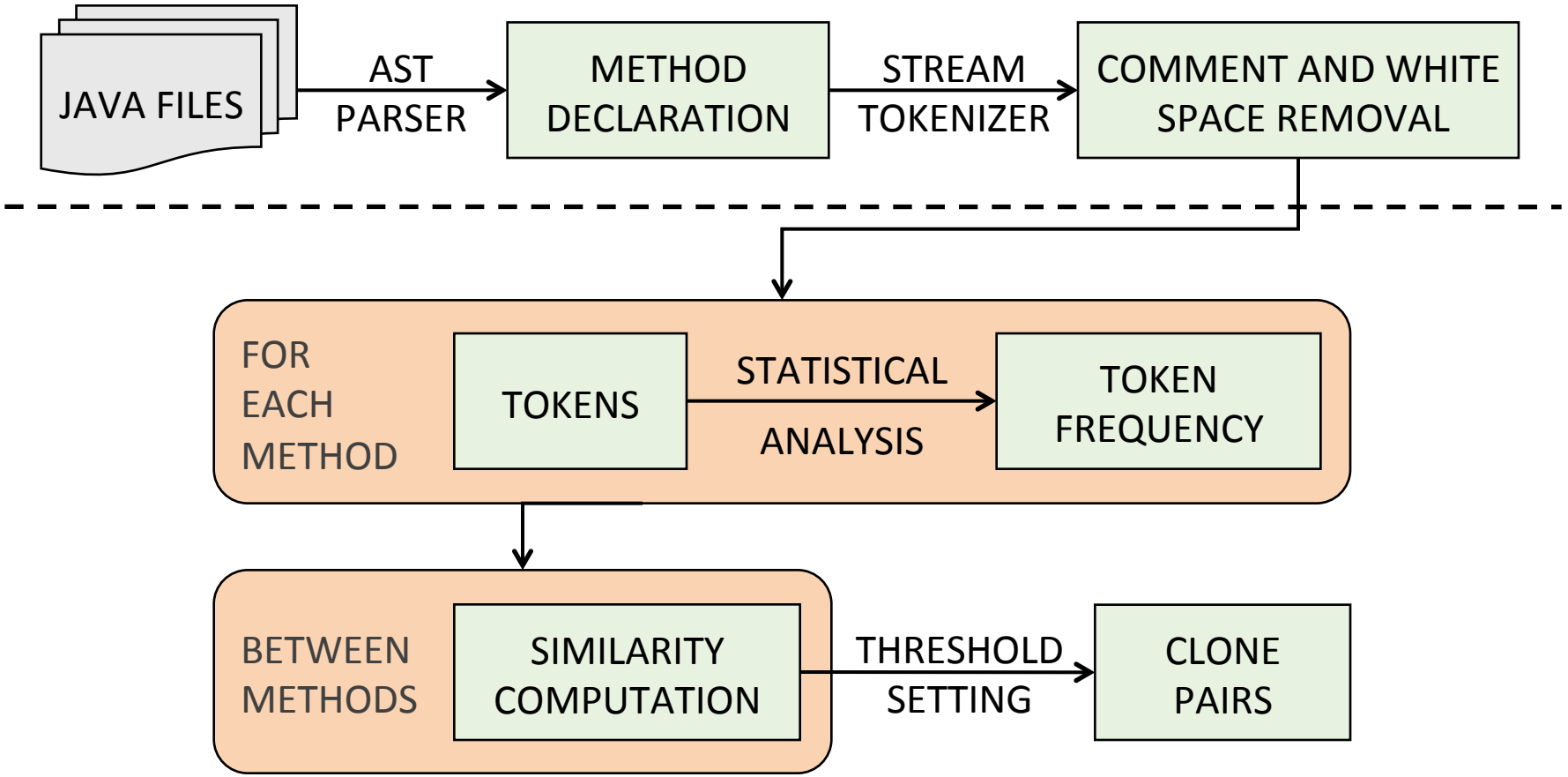
- Problem Definition and Solution Design
- Approaches of Solving Challenges
- Experiment Results and Evaluation
- Things We Learn

Problem Definition and Solution Design

- Goal
  - Design and implement STCD (Statistical-based Clone Detection) tool to detect the Type 1, 2, 3 code clones between methods based on tokens
- Solution Design

Problem Definition and Solution Design

- Overall Current Project Diagram



Problem Definition and Solution Design

- Similarity Computation
  - Method Similarity
  - 9-Dimentional Vector
    - X = <X1, X2, X3, X4, X5, X6, X7, X8, X9>
    - Y = <Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9>

    - Sim(X1, Y1) = bigram(X1, Y1)
    - Sim(X2, Y2) = 1 / 0

    - Sim(X_{else}, Y_{else}) = 1 / 1 + Distance(X_{else}, Y_{else})
    - Distance (X_{else}, Y_{else}): Euclidean Distance

    - Sim(X, Y) = Sum(wi * Sim(Xi, Yi)) (i = 1, 2, …, 9)

Vector
methodPara
methodType
tokenListNum
tokenListType
tokenListKeyword
tokenListMarker
tokenListOperator
tokenListOther1
tokenListOther2

Problem Definition and Solution Design

- Threshold Setting 1
  - For Variables: DrawPointLine vs. PointLineDraw
  - tokenThreshold for bigram similarity
  - e.g. variableSimilarity > 0.7
- Threshold Setting 2
  - detectThreshold for method similarity
  - e.g. methodsSimilarity > 0.5
- Threshold Setting 3
  - lineThreshold for method lines
  - e.g. endLineNum - startLineNum > 7

Approaches of Solving Challenges

- ASTParser
  - Excessive time cost
  - Improve the method parsing process
  - e.g. Regular expression

- Manual weights and threshold
  - 9-Dimentional (w1, …, w9)
  - Machine Learning Techniques (Training Data)
  - e.g. Multilayer perceptron (MLP)

Approaches of Solving Challenges

- Variables Comparison
  - Bigram Algorithm
- Data Collection
  - Training Data for Machine Learning
  - Test Data for results comparison
- Results Comparison
  - Precision & Recall
- UI Development
  - Java WindowBuilder

- I’ll start from here…

Experiment Results and Evaluation

- Source: https://github.com/CSCC5704
  - CodeClone.java
    - Detect cloned code in java files
  - MethodSimilarity.java
    - Calculate the similarity of two methods
  - BiGramSimilarity.java
    - Calculate the similarity of two strings by using bi-gram algorithm
  - ASTParserTool.java
    - Use JDT ASTParser to parse the java source code into methods
  - MethodTokenizerTool.java
    - Tokenize method body and get the frequency of tokens
  - MethodList.java, MethodVector.java
  - TokenList.java, TokenVector.java, …

Experiment Results and Evaluation

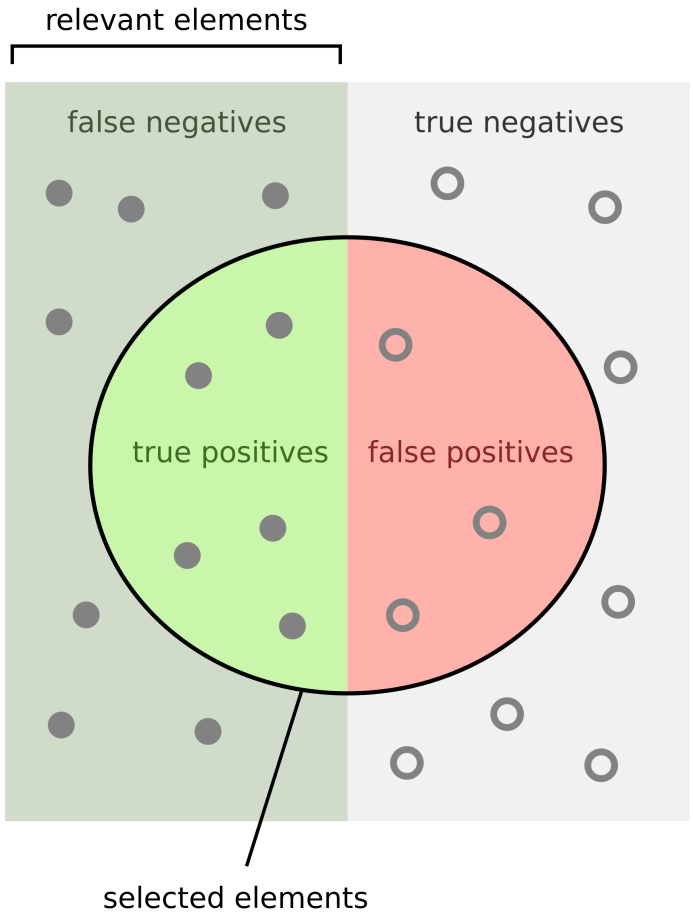
- Our UI (Add a picture here or demonstrate)

Experiment Results and Evaluation

- TrainingData: SWT
- TestData: SWT
- Detection of Software Clones
  - http://www.bauhaus-stuttgart.de/clones/

Experiment Results and Evaluation

- Evaluation: Precision and Recall



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

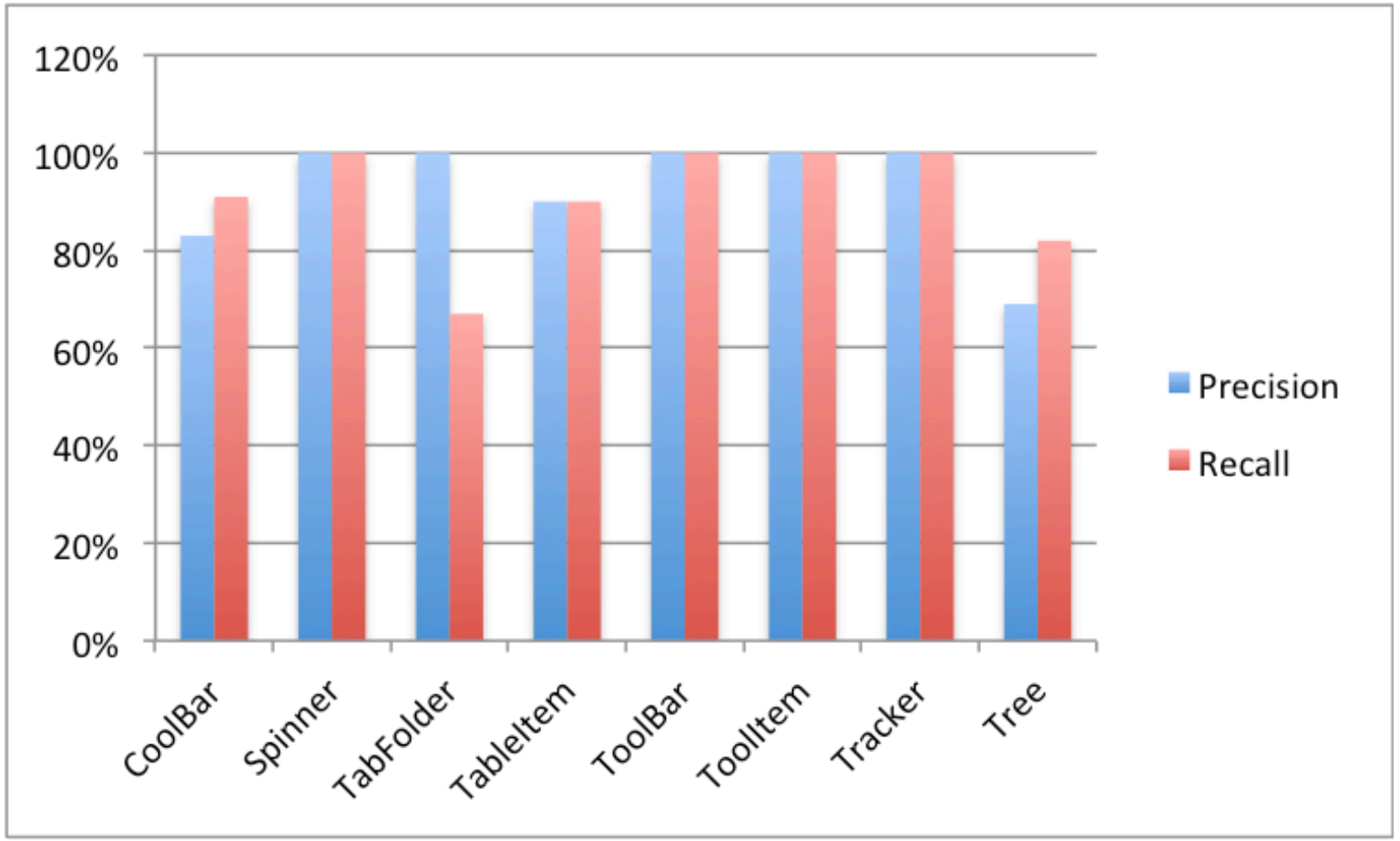
Experiment Results and Evaluation

- Evaluation: Precision and Recall (manual selection)

	TP+FN (actual clones)	TP+FP (clones we got)	TP (correct ones)	Precision	Recall
Button	0	4	0	0	
CoolBar	11	12	10	83%	91%
Menu	1	0	0		0
Spinner	2	2	2	100%	100%
TabFolder	3	2	2	100%	67%
TableItem	20	20	18	90%	90%
ToolBar	4	4	4	100%	100%
ToolItem	3*	3	3	100%	100%
Tracker	1	1	1	100%	100%
Tree	11	13	9	69%	82%

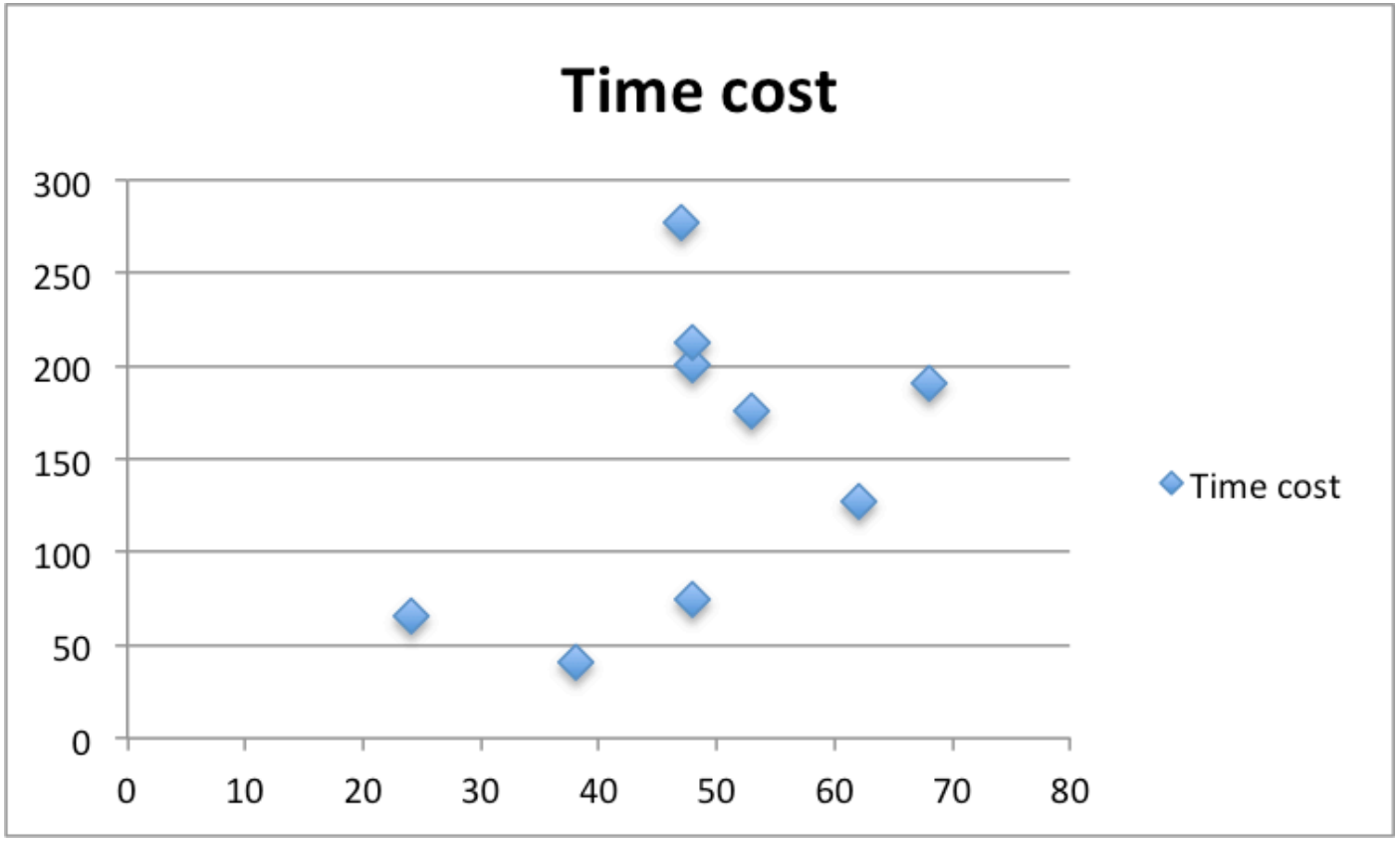
Experiment Results and Evaluation

- Evaluation: Precision and Recall



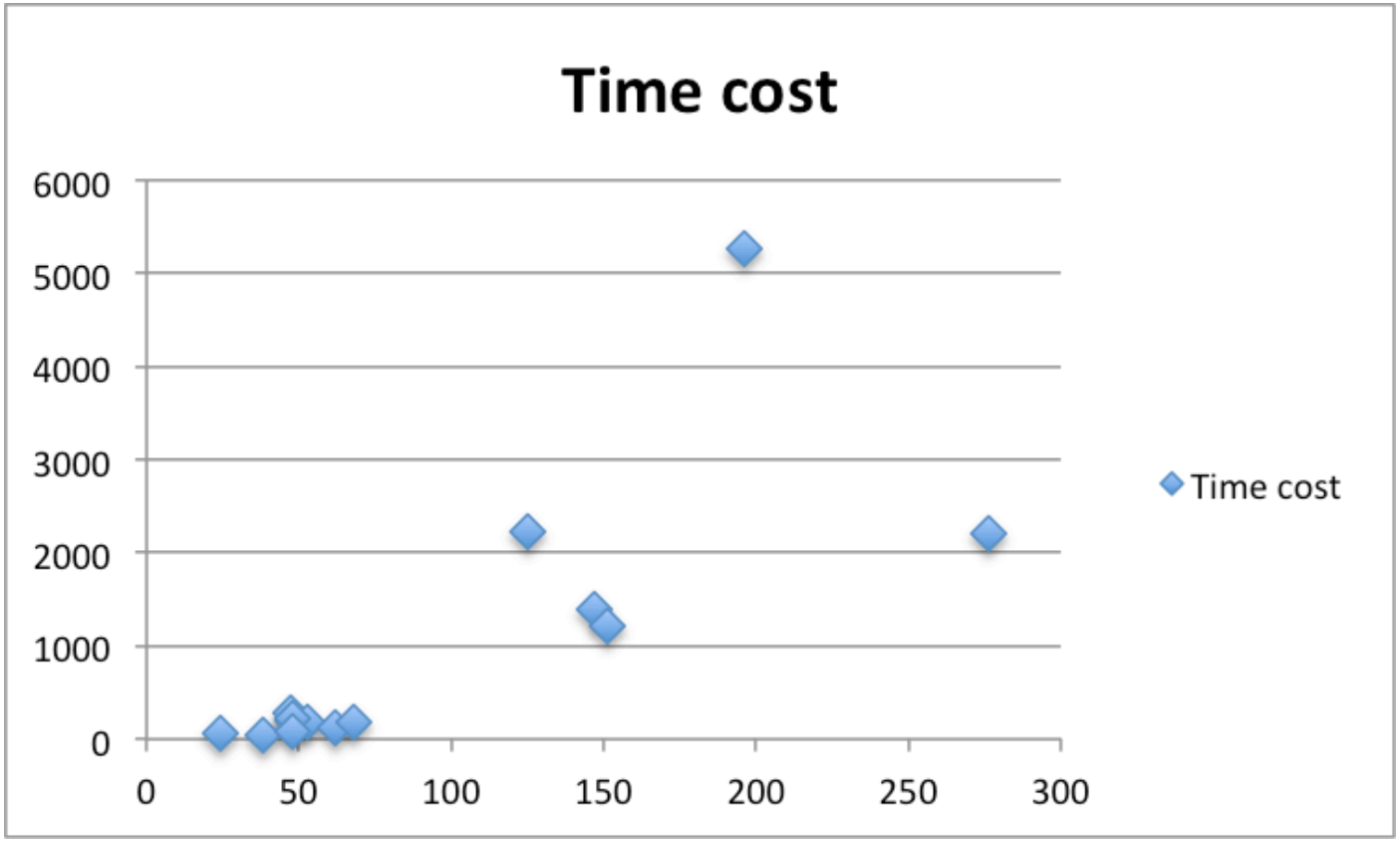
Experiment Results and Evaluation

- Evaluation: Time Cost (in terms of # of methods)



Experiment Results and Evaluation

- Evaluation: Time Cost (with several large files)



Things We Learn

- ASTParserTool
- Bigram Algorithm

Thank you !