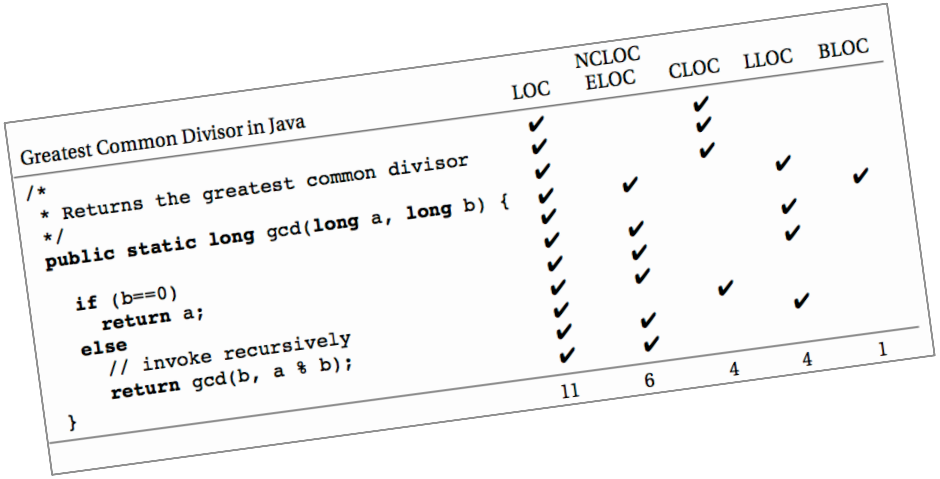
MaLTeSQuE2017, Feb 21st, 2017, Klagenfurt

Using ML to Design a Flexible LOC Counter



Greatest Common Divisor in Java	LOC	NCLOC ELOC	CLOC	LLOC	BLOC
`/*`	✓		✓		
`* Returns the greatest common divisor`	✓		✓		
`*/`	✓		✓		
`public static long gcd(long a, long b) {`	✓	✓		✓	
	✓				✓
`if (b==0)`	✓	✓		✓	
`return a;`	✓	✓		✓	
`else`	✓	✓			
`// invoke recursively`	✓		✓		
`return gcd(b, a % b);`	✓	✓		✓	
`}`	✓	✓			
	11	6	4	4	1

ERICSSON

UNIVERSITY OF GOTHENBURG

CHALMERS

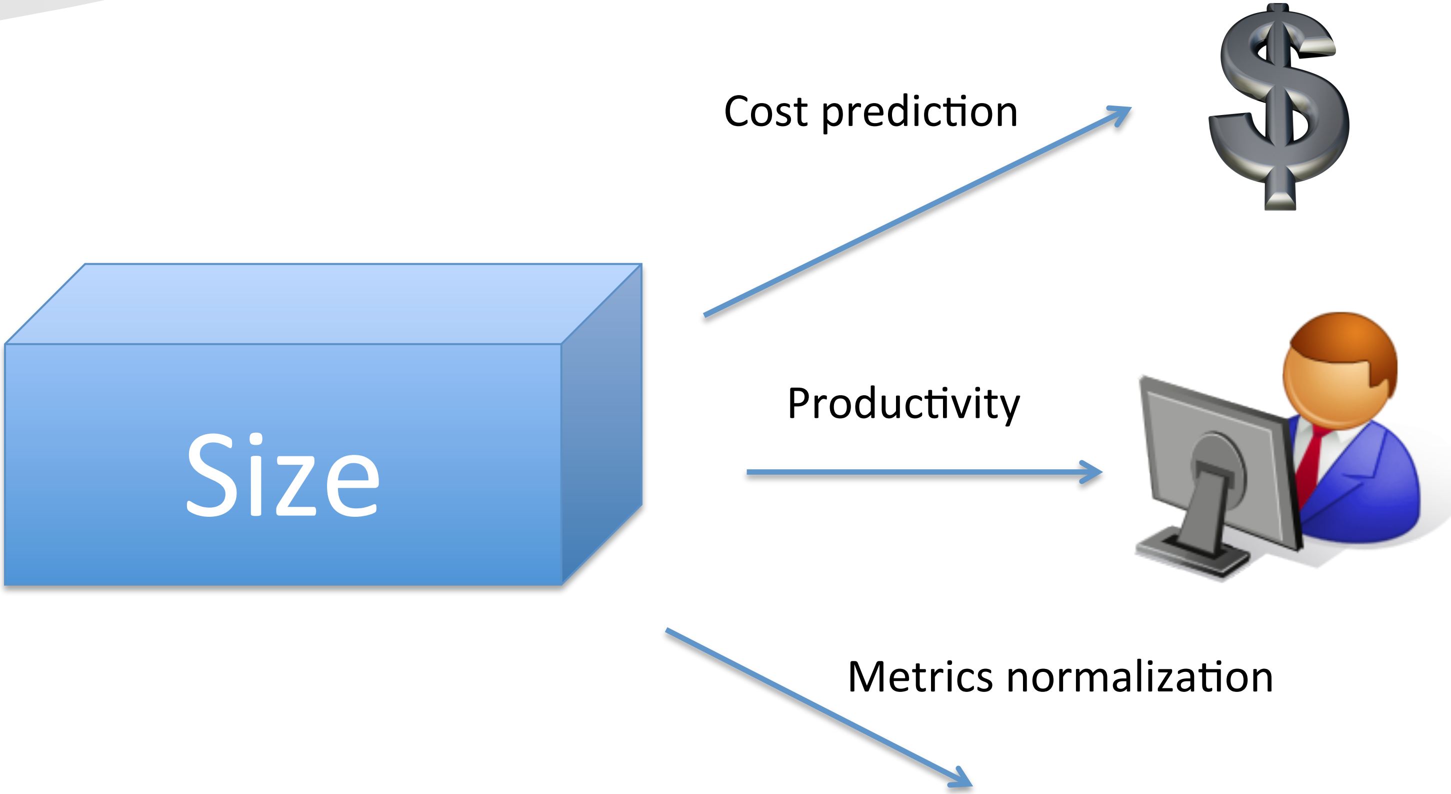
POLITECHNIKA POZNAŃSKA

POZNAN UNIVERSITY OF TECHNOLOGY

Mirosław Ochodek
Miroslaw Staron
Dominik Bargowski
Wilhelm Meding
Regina Hebig

Workshop on Machine Learning Techniques for Software Quality Evaluation

Software size



Size

Cost prediction

Productivity

Metrics normalization

$$\text{Defects density} = \frac{\#\text{Defects}}{\text{Size}}$$

The Problem

Improving Measurement Certainty by Using Calibration to Find Systematic Measurement Error – A Case of Lines-of-Code Measure

Miroslaw Staron[1], Darko Durisic[2], and Rakesh Rana[1]

[1] Computer Science and Engineering, University of Gothenburg, Sweden
miroslaw.staron/rakesh.rana@gu.se,
[2] Volvo Car Group, Sweden
darko.durisic@volvocars.com

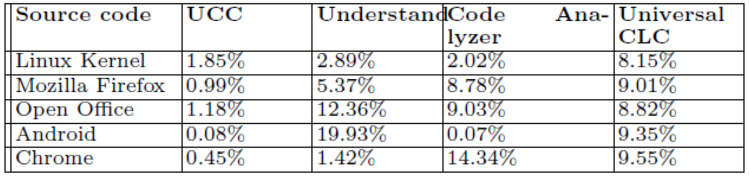
Abstract. Base measures such as the number of lines-of-code are often used to make predictions about such phenomena as project effort, product quality or maintenance effort. However, quite often we rely on the measurement instruments where the exact algorithm for calculating the value of the measure is not known. The objective of our research is to explore how we can increase the certainty of base measures in software engineering. We conduct a benchmarking study where we use four measurement instruments for lines-of-code measurement with unknown certainty to measure five code bases. Our results show that we can adjust the measurement values by as much as 20% knowing the systematic error of the tool. We conclude that calibrating the measurement instruments can significantly contribute to increased accuracy in measurement processes in software engineering. This will impact the accuracy of predictions (e.g. of effort in software projects) and therefore increase the cost-efficiency of software engineering processes.

1 Introduction

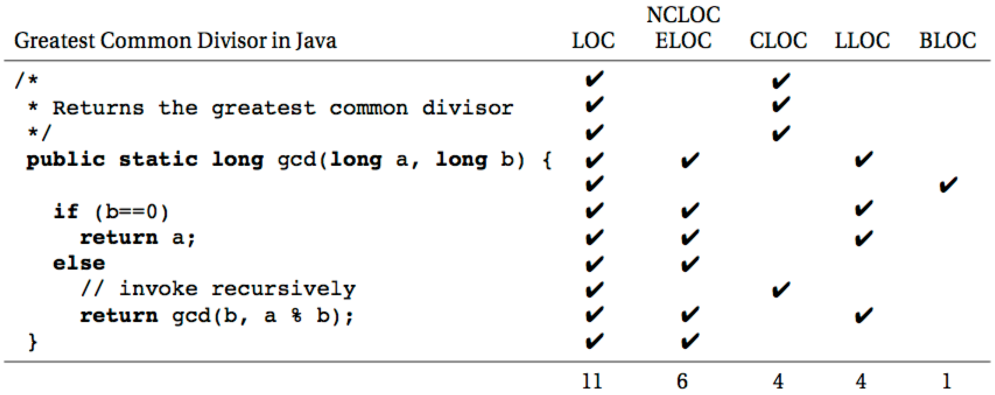
With the introduction of the measurement information model in the international ISO/IEC 15939 standard for measurement processes the discipline of software engineering evolved from discussing metrics in general to categorizing them into three categories – base measures, derived measures and indicators. The use of base measures is fundamental for the construction of derived measures and indicators. The base measures are also the types of measures which are collected directly and are a result of a measurement method. In many cases this measurement method is an automated algorithm (e.g. a script) which we can refer to as the *measurement instrument* which quantifies an attribute of interest into a number.

Since in software engineering we do not have reference measurement etalons as we do in other disciplines (e.g. kilogram or meter for physics), we often rely on arbitrary definitions of the base quantities. One of such quantities is the size of programs measured as the number of lines of code. Even though the number of lines of code of a given program is a deterministic and fully quantifiable

Four tools
Error (vs. median)
up to ~20%



Source code	UCC	Understand	Code Analyzer	Universal CLC
Linux Kernel	1.85%	2.89%	2.02%	8.15%
Mozilla Firefox	0.99%	5.37%	8.78%	9.01%
Open Office	1.18%	12.36%	9.03%	8.82%
Android	0.08%	19.93%	0.07%	9.35%
Chrome	0.45%	1.42%	14.34%	9.55%



Greatest Common Divisor in Java	LOC	NCLOC ELOC	CLOC	LLOC	BLOC
`/*`	✔		✔		
`* Returns the greatest common divisor`	✔		✔		
`*/`	✔		✔		
`public static long gcd(long a, long b) {`	✔	✔		✔	
	✔				✔
`if (b==0)`	✔	✔		✔	
`return a;`	✔	✔		✔	
`else`	✔	✔			
`// invoke recursively`	✔		✔		
`return gcd(b, a % b);`	✔	✔		✔	
`}`	✔	✔			
	11	6	4	4	1

Output: 2512 LOC

Introduces (unknown) measurement error, problems with reliability of the measurement, difficulties in measuring multi-language code base…

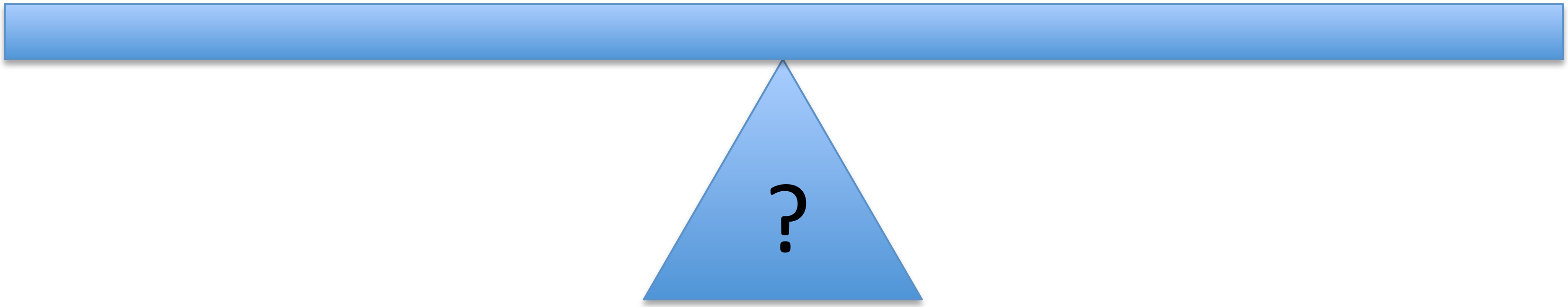
Potential solutions

A tool based on Programming Language (PL) parsers

- *Explicitly known rules for counting that can be somehow formulated*
- 100% accurate according to the rules
- Requires implementation for each PL
- Can be also implemented to allow for some configuration of rules (however, probably somehow limited)

A machine learning (ML) approach

- *It is difficult to explicitly define the rules (either not known or too complex)*
- Learns from examples (require training set)
- Classification error depending on the quality of training set
- Doesn't require new implementation for new language (however, may require a new training set)



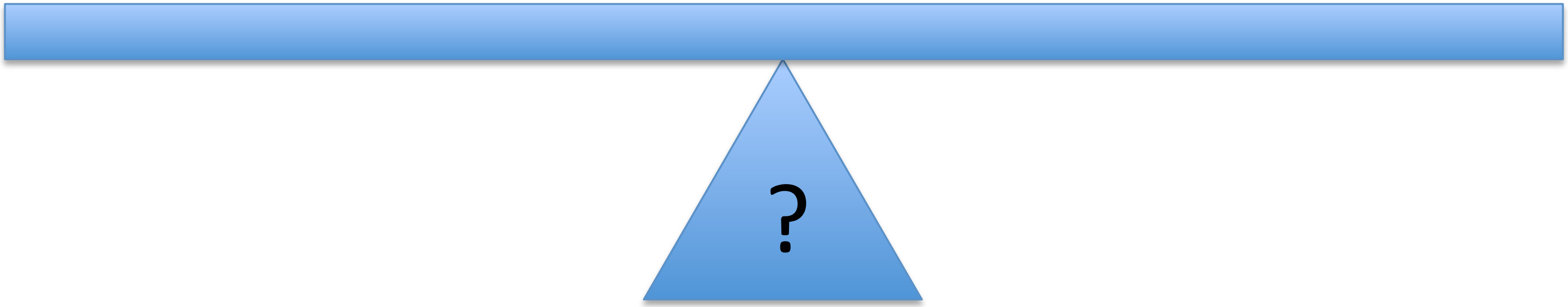
Potential solutions

A tool based on Programming Language (PL) parsers

- *Explicitly known rules for counting that can be somehow formulated*
- 100% accurate according to the rules
- Requires implementation for each PL
- Can be also implemented to allow for some configuration of rules (however, probably somehow limited)

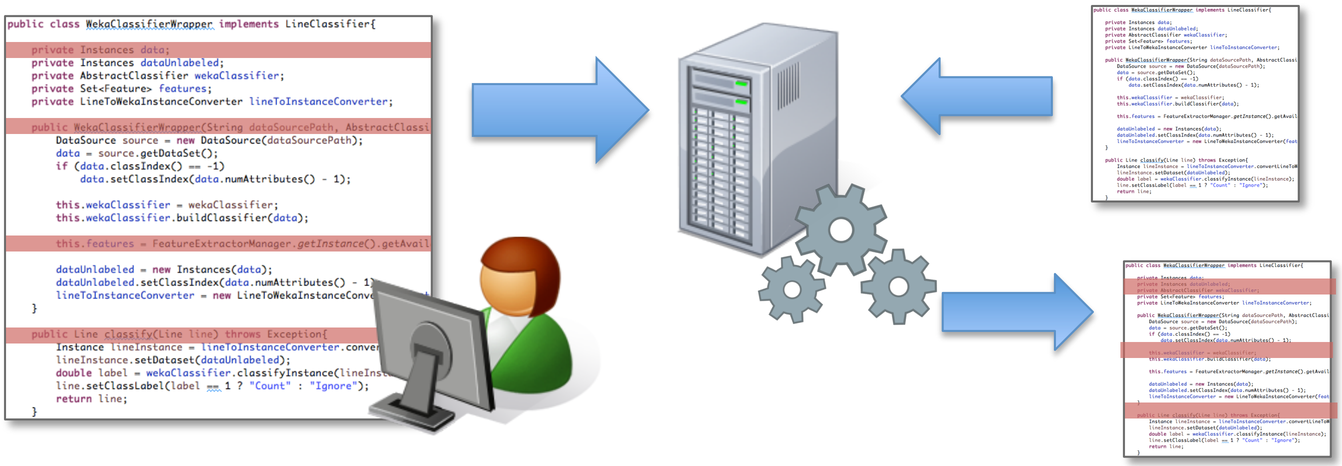
A machine learning (ML) approach

- *It is difficult to explicitly define the rules (either not known or too complex)*
- Learns from examples (require training set)
- Classification error depending on the quality of training set
- Doesn't require new implementation for new language (however, may require a new training set)



Idea of the solution

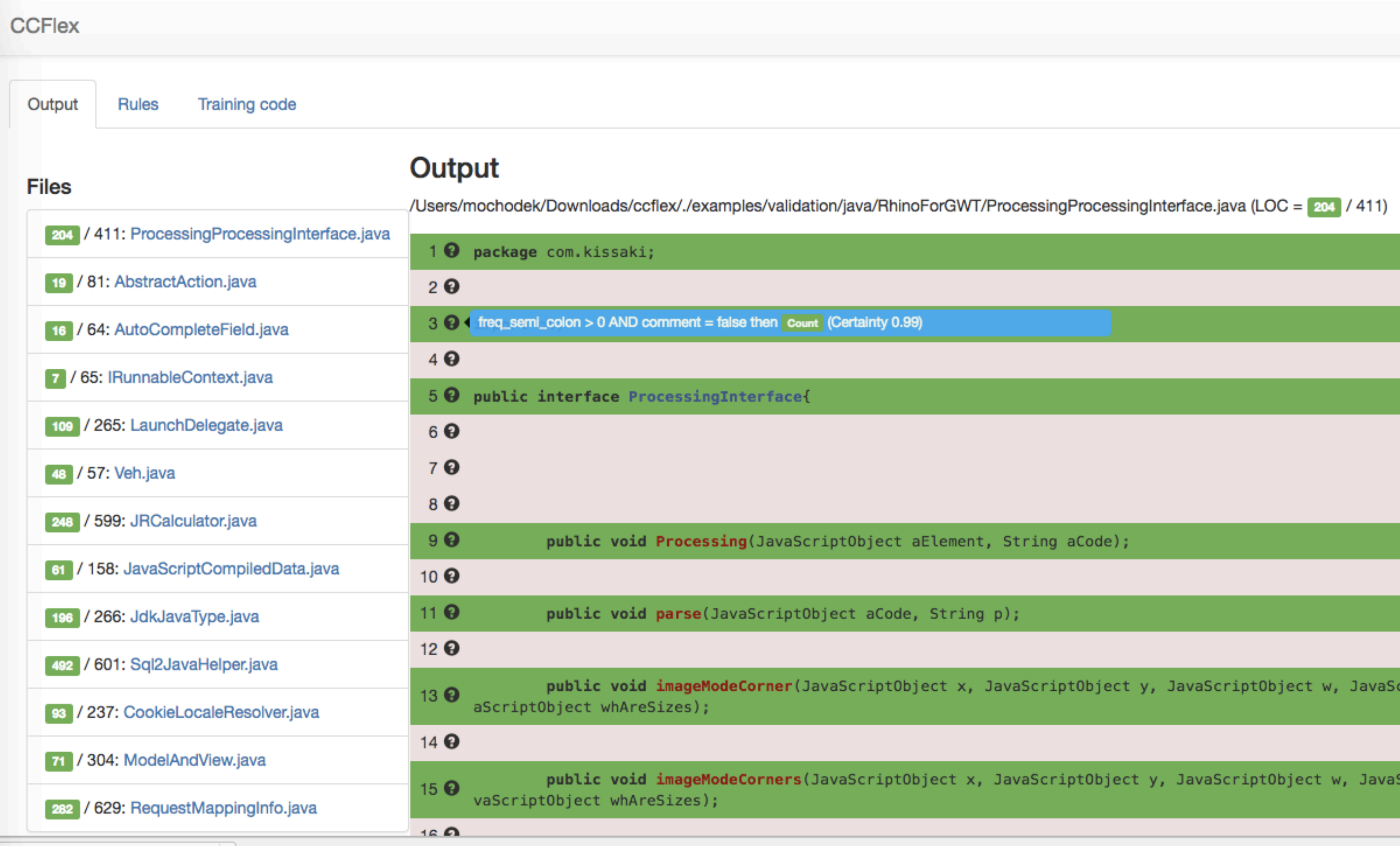
- Flexible lines of code counter (CCFlex)
  - A user teaches the tool which lines should be counted based on a sample (a training set)



```
public class WekaClassifierWrapper implements LineClassifier{

    private Instances data;
    private Instances dataUnlabeled;
    private AbstractClassifier wekaClassifier;
    private Set<Feature> features;
    private LineToWekaInstanceConverter lineToInstanceConverter;

    public WekaClassifierWrapper(String dataSourcePath, AbstractClassi
        DataSource source = new DataSource(dataSourcePath);
        data = source.getDataSet();
        if (data.classIndex() == -1)
            data.setClassIndex(data.numAttributes() - 1);

        this.wekaClassifier = wekaClassifier;
        this.wekaClassifier.buildClassifier(data);

        this.features = FeatureExtractorManager.getInstance().getAvail

        dataUnlabeled = new Instances(data);
        dataUnlabeled.setClassIndex(data.numAttributes() - 1)
        lineToInstanceConverter = new LineToWekaInstanceConve
    }

    public Line classify(Line line) throws Exception{
        Instance lineInstance = lineToInstanceConverter.conver
        lineInstance.setDataset(dataUnlabeled);
        double label = wekaClassifier.classifyInstance(lineInst
        line.setClassLabel(label == 1 ? "Count" : "Ignore");
        return line;
    }
```

10 LOC
Justification

Idea of the solution



Files

- 204 / 411: ProcessingProcessingInterface.java
- 19 / 81: AbstractAction.java
- 16 / 64: AutoCompleteField.java
- 7 / 65: IRunnableContext.java
- 109 / 265: LaunchDelegate.java
- 48 / 57: Veh.java
- 248 / 599: JRCalculator.java
- 61 / 158: JavaScriptCompiledData.java
- 196 / 266: JdkJavaType.java
- 492 / 601: Sql2JavaHelper.java
- 93 / 237: CookieLocaleResolver.java
- 71 / 304: ModelAndView.java
- 282 / 629: RequestMappingInfo.java

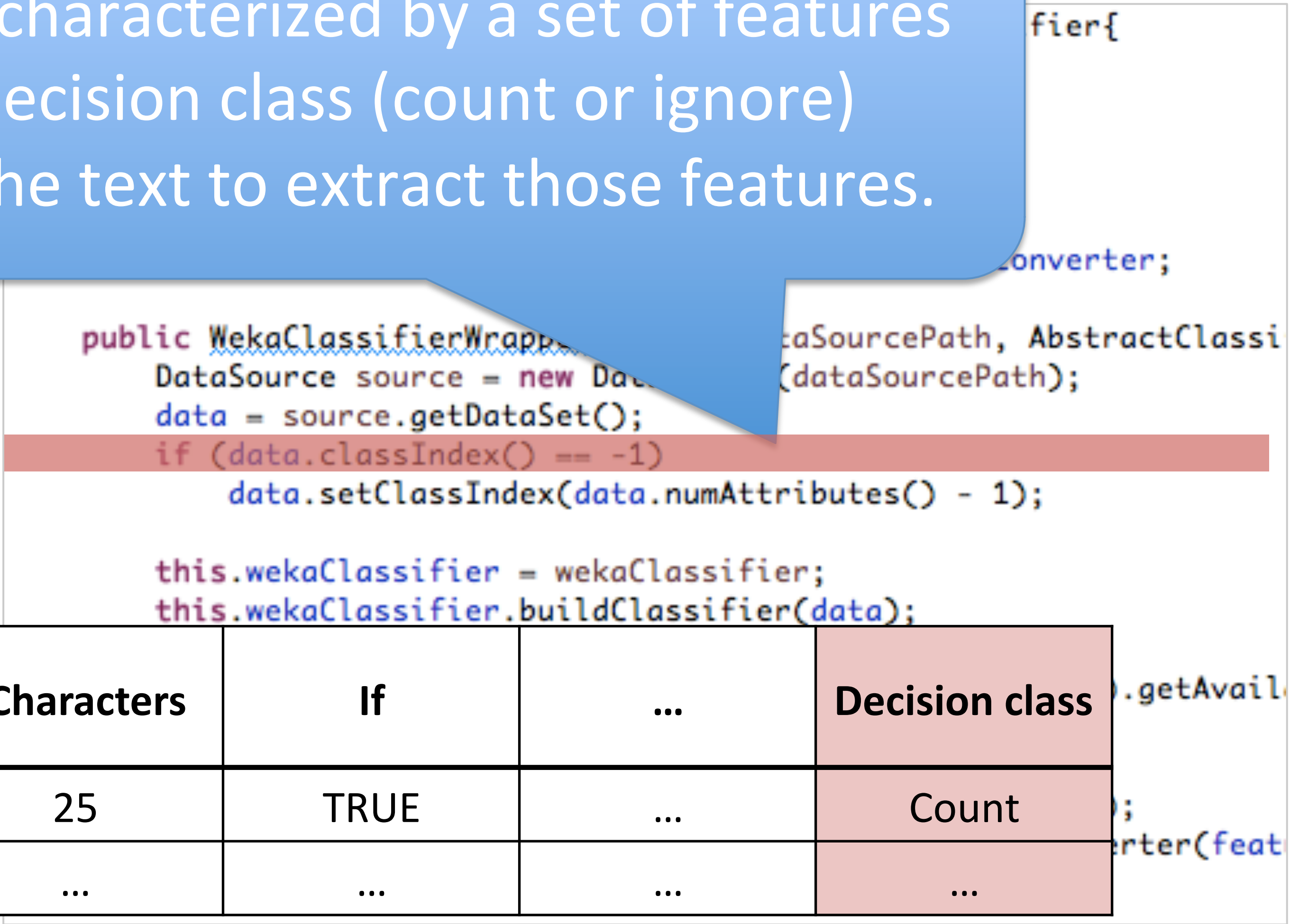
Output

/Users/mochodek/Downloads/ccflex/./examples/validation/java/RhinoForGWT/ProcessingProcessingInterface.java (LOC = 204 / 411)

```
package com.kissaki;


freq_semi_colon > 0 AND comment = false then Count (Certainty 0.99)

public interface ProcessingInterface{



        public void Processing(JavaScriptObject aElement, String aCode);

        public void parse(JavaScriptObject aCode, String p);

        public void imageModeCorner(JavaScriptObject x, JavaScriptObject y, JavaScriptObject w, JavaSc
aScriptObject whAreSizes);

        public void imageModeCorners(JavaScriptObject x, JavaScriptObject y, JavaScriptObject w, Java
vaScriptObject whAreSizes);
```

Feature acquisition

Each line is characterized by a set of features and its decision class (count or ignore)
We parse the text to extract those features.

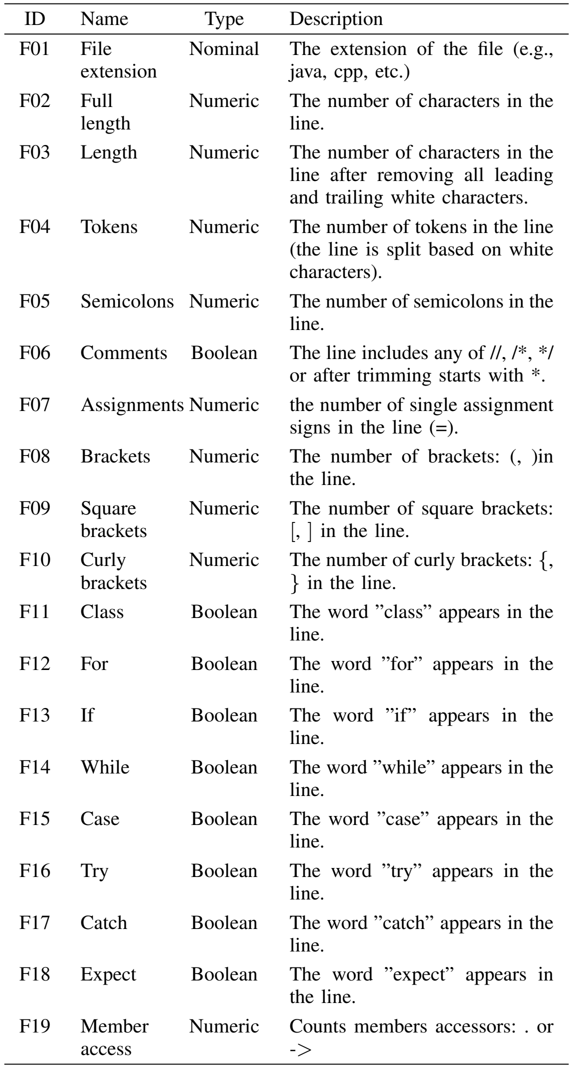


```
                                                   fier{



                                                   onverter;

    public WekaClassifierWrapp            aSourcePath, AbstractClassi
        DataSource source = new Da        (dataSourcePath);
        data = source.getDataSet();
        if (data.classIndex() == -1)
            data.setClassIndex(data.numAttributes() - 1);

        this.wekaClassifier = wekaClassifier;
        this.wekaClassifier.buildClassifier(data);
                                                   .getAvail

                                                   );
                                                   rter(feat
```

File type	#Characters	If	…	Decision class
java	25	TRUE	…	Count
…	…	…	…	…

Feature acquisition

- **Plain text (F01-F04):**
  - File extension
  - Full and trimmed length (characters)
  - Tokens
- **Programming language (F05-F19):**
  - Assignment,
  - Brackets,
  - Class,
  - Comment,
  - Semicolons,
  - …



ID	Name	Type	Description
F01	File extension	Nominal	The extension of the file (e.g., java, cpp, etc.)
F02	Full length	Numeric	The number of characters in the line.
F03	Length	Numeric	The number of characters in the line after removing all leading and trailing white characters.
F04	Tokens	Numeric	The number of tokens in the line (the line is split based on white characters).
F05	Semicolons	Numeric	The number of semicolons in the line.
F06	Comments	Boolean	The line includes any of //, /*, */ or after trimming starts with *.
F07	Assignments	Numeric	the number of single assignment signs in the line (=).
F08	Brackets	Numeric	The number of brackets: (,)in the line.
F09	Square brackets	Numeric	The number of square brackets: [,] in the line.
F10	Curly brackets	Numeric	The number of curly brackets: {, } in the line.
F11	Class	Boolean	The word ”class” appears in the line.
F12	For	Boolean	The word ”for” appears in the line.
F13	If	Boolean	The word ”if” appears in the line.
F14	While	Boolean	The word ”while” appears in the line.
F15	Case	Boolean	The word ”case” appears in the line.
F16	Try	Boolean	The word ”try” appears in the line.
F17	Catch	Boolean	The word ”catch” appears in the line.
F18	Expect	Boolean	The word ”expect” appears in the line.
F19	Member access	Numeric	Counts members accessors: . or ->

Feature acquisition

- **Bag of words approach (automatic)**
  - Tokenize: ()[]{}!@#$%^&*-=;:’”\|`~,.<>/?
  - Treat split character as a token
  - Calculate thresholds:
    - Frequencies of tokens in the code base (min. 5)
    - % of files a token is present in (min. 25%)
  - If thresholds are met:
    - F_i: the number of times the token$_i$ occurs in a line

Preliminary validation

- **RQ1:** What level of prediction quality can be achieved by the proposed approach?
- **RQ2:** How the automatic features acquisition affects the classification quality?
- **RQ3:** How the choice of classification algorithm affects the classification quality?

Code databases

- 2402 physical lines of code in total
  - – Eclipse: 475 LOC,
  - – Jasper Reports 757 LOC,
  - – Spring MVC: 1170 LOC
- **ELOC** (Count 1492 / Ignore 910)
- **Subjective** (Count 1237, Ignore 1165)

Validation schemes

10 x 10-fold cross-validation (18 schemes)

- two datasets
  - ELOC
  - Subjective;
- three feature sets
  - All: F01–F19 and acquired automatically;
  - Auto: F01–F04 and acquired automatically;
  - Predefined: F01–F19;
- three classification algorithms (PART, JRip, J48).

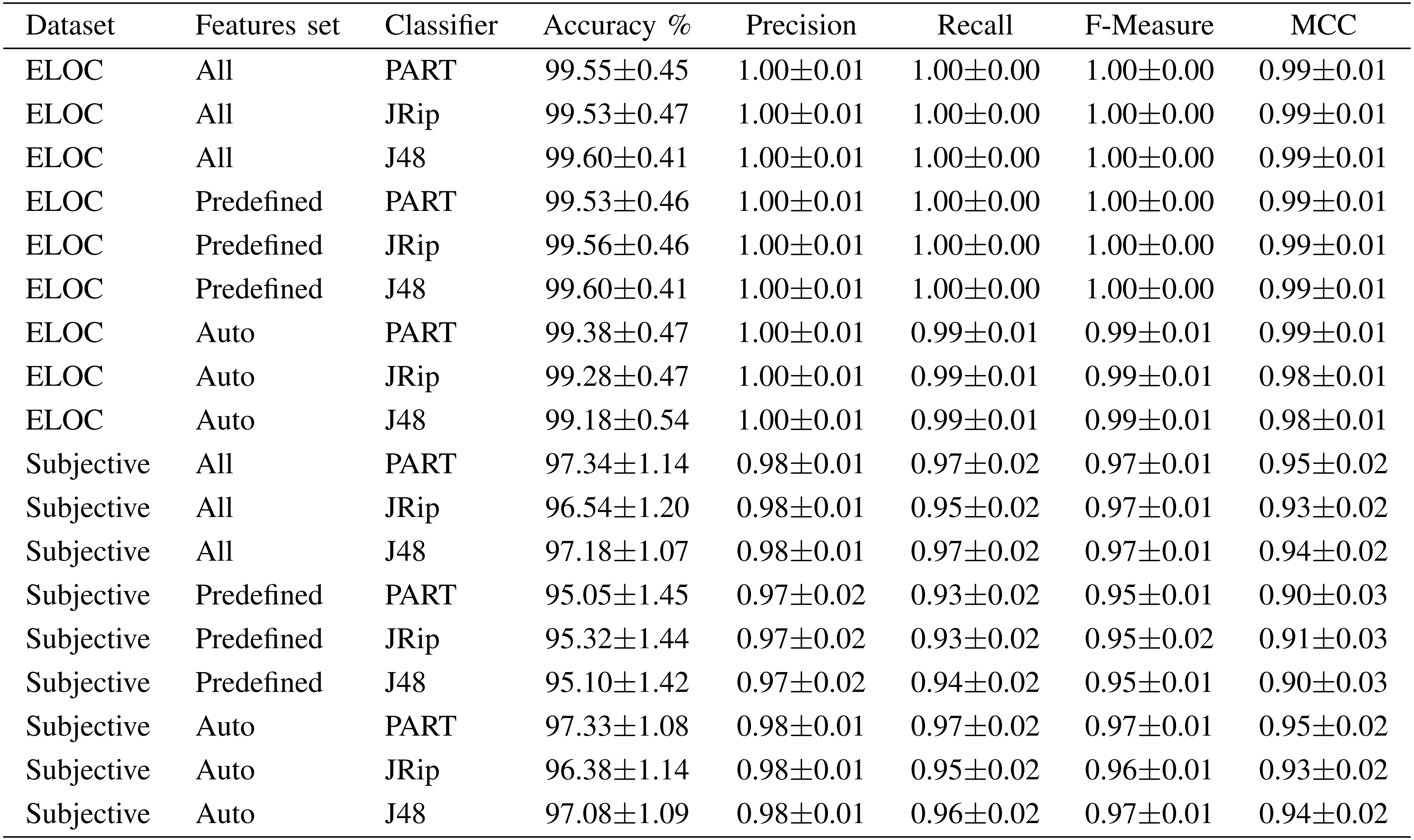
Prediction quality measures

- Accuracy
- Precision
- Recall
- F-score
- Matthews Correlation Coefficient (MCC)

Results

RQ1: What level of prediction quality can be achieved by the proposed approach?

Results



Dataset	Features set	Classifier	Accuracy %	Precision	Recall	F-Measure	MCC
ELOC	All	PART	99.55±0.45	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	All	JRip	99.53±0.47	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	All	J48	99.60±0.41	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	Predefined	PART	99.53±0.46	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	Predefined	JRip	99.56±0.46	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	Predefined	J48	99.60±0.41	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	Auto	PART	99.38±0.47	1.00±0.01	0.99±0.01	0.99±0.01	0.99±0.01
ELOC	Auto	JRip	99.28±0.47	1.00±0.01	0.99±0.01	0.99±0.01	0.98±0.01
ELOC	Auto	J48	99.18±0.54	1.00±0.01	0.99±0.01	0.99±0.01	0.98±0.01
Subjective	All	PART	97.34±1.14	0.98±0.01	0.97±0.02	0.97±0.01	0.95±0.02
Subjective	All	JRip	96.54±1.20	0.98±0.01	0.95±0.02	0.97±0.01	0.93±0.02
Subjective	All	J48	97.18±1.07	0.98±0.01	0.97±0.02	0.97±0.01	0.94±0.02
Subjective	Predefined	PART	95.05±1.45	0.97±0.02	0.93±0.02	0.95±0.01	0.90±0.03
Subjective	Predefined	JRip	95.32±1.44	0.97±0.02	0.93±0.02	0.95±0.02	0.91±0.03
Subjective	Predefined	J48	95.10±1.42	0.97±0.02	0.94±0.02	0.95±0.01	0.90±0.03
Subjective	Auto	PART	97.33±1.08	0.98±0.01	0.97±0.02	0.97±0.01	0.95±0.02
Subjective	Auto	JRip	96.38±1.14	0.98±0.01	0.95±0.02	0.96±0.01	0.93±0.02
Subjective	Auto	J48	97.08±1.09	0.98±0.01	0.96±0.02	0.97±0.01	0.94±0.02

Results

Dataset	Features set	Classifier	Accuracy %	Precision	Recall	F-Measure	MCC
ELOC	All	PART	99.55±0.45	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	All	JRip	99.53±0.47	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	All	J48	99.60±0.41	1.00	[illegible]	[illegible]	0.99±0.01
ELOC	Predefined	PART	99.53±0.46	1.	[illegible]	[illegible]	0.99±0.01
ELOC	Predefined	JRip	99.56±0.46	1.00	[illegible]	[illegible]	0.99±0.01
ELOC	Predefined	J48	99.60±0.41	1.00	[illegible]	[illegible]	0.99±0.01
ELOC	Auto	PART	99.38±0.47	1.00	[illegible]	[illegible]	0.99±0.01
ELOC	Auto	JRip	99.28±0.47	1.00±0.01	0.99±0.01	0.99±0.01	0.98±0.01
ELOC	Auto	J48	99.18±0.54	1.00±0.01	0.99±0.01	0.99±0.01	0.98±0.01
Subjective	All	PART	97.34±1.14	0.98±0.01	0.97±0.02	0.97±0.01	0.95±0.02
Subjective	All	JRip	96.54±1.20	0.98±0.01	0.95±0.02	0.97±0.01	0.93±0.02
Subjective	All	J48	97.18±1.07	0.98±0.01	0.97±0.02	0.97±0.01	0.94±0.02
Subjective	Predefined	PART	95.05±1.45	0.97±0.02	0.93±0.02	0.95±0.01	0.90±0.03
Subjective	Predefined	JRip	95.32±1.44	0.97±0.02	0.93±0.02	0.95±0.02	0.91±0.03
Subjective	Predefined	J48	95.10±1.42	0.97±0.02	0.94±0.02	0.95±0.01	0.90±0.03
Subjective	Auto	PART	97.33±1.08	0.98±0.01	0.97±0.02	0.97±0.01	0.95±0.02
Subjective	Auto	JRip	96.38±1.14	0.98±0.01	0.95±0.02	0.96±0.01	0.93±0.02
Subjective	Auto	J48	97.08±1.09	0.98±0.01	0.96±0.02	0.97±0.01	0.94±0.02

Very high accuracy:
95.05 - 99.60%

Higher accuracy for ELOC

Results

Dataset	Features set	Classifier	Accuracy %	Precision	Recall	F-Measure	MCC
ELOC	All	PART	99.55±0.45	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	All	JRip	99.53±0.47	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	All	J48	99.60±0.41	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC			3±0.46	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC			±0.46	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC			[illegible]	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC			8±0.47	1.00±0.01	0.99±0.01	0.99±0.01	0.99±0.01
ELOC			28±0.47	1.00±0.01	0.99±0.01	0.99±0.01	0.98±0.01
ELOC			8±0.54	1.00±0.01	0.99±0.01	0.99±0.01	0.98±0.01
Subjecti			4±1.14	0.98±0.01	0.97±0.02	0.97±0.01	0.95±0.02
Subjective	All	JRip	96.54±1.20	0.98±0.01	0.95±0.02	0.97±0.01	0.93±0.02
Subjective	All	J48	97.18±1.07	0.98±0.01	0.97±0.02	0.97±0.01	0.94±0.02
Subjective	Predefined	PART	95.05±1.45	0.97±0.02	0.93±0.02	0.95±0.01	0.90±0.03
Subjective	Predefined	JRip	95.32±1.44	0.97±0.02	0.93±0.02	0.95±0.02	0.91±0.03
Subjective	Predefined	J48	95.10±1.42	0.97±0.02	0.94±0.02	0.95±0.01	0.90±0.03
Subjective	Auto	PART	97.33±1.08	0.98±0.01	0.97±0.02	0.97±0.01	0.95±0.02
Subjective	Auto	JRip	96.38±1.14	0.98±0.01	0.95±0.02	0.96±0.01	0.93±0.02
Subjective	Auto	J48	97.08±1.09	0.98±0.01	0.96±0.02	0.97±0.01	0.94±0.02

Very high Precision and Recall (0.93-1.00)

Slight preference towards Precision

Small standard deviations

Results

RQ2: How the automatic features acquisition affects the classification quality?

Results

Dataset	Features set	Classifier	Accuracy %	Precision	Recall	F-Measure	MCC
ELOC	All	PART	99.55±0.45	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	All	JRip	99.53±0.47	1.			0.99±0.01
ELOC	All	J48	99.60±0.41	1.			0.99±0.01
ELOC	Predefined	PART	99.53±0.46	1.			0.99±0.01
ELOC	Predefined	JRip	99.56±0.46	1.			0.99±0.01
ELOC	Predefined	J48	99.60±0.41	1.			0.99±0.01
ELOC	Auto	PART	99.38±0.47	1.00±0.01	0.99±0.01	0.99±0.01	0.99±0.01
ELOC	Auto	JRip	99.28±0.47	1.00±0.01	0.99±0.01	0.99±0.01	0.98±0.01
ELOC	Auto	J48	99.18±0.54	1.00±0.01	0.99±0.01	0.99±0.01	0.98±0.01
Subjective	All	PART	97.34±1.14	0.98±0.01	0.97±0.02	0.97±0.01	0.95±0.02
Subjective	All	JRip	96.54±1.20	0.98±0.01	0.95±0.02	0.97±0.01	0.93±0.02
Subjective	All	J48	97.18±1.07	0.98±0.01	0.97±0.02	0.97±0.01	0.94±0.02
Subjective	Predefined	PART	95.05±1.45	0.97±0.02	0.93±0.02	0.95±0.01	0.90±0.03
Subjective	Predefined	JRip	95.32±1.44	0.97±0.02	0.93±0.02	0.95±0.02	0.91±0.03
Subjective	Predefined	J48	95.10±1.42	0.97			0.90±0.03
Subjective	Auto	PART	97.33±1.08	0.98			0.95±0.02
Subjective	Auto	JRip	96.38±1.14	0.98			0.93±0.02
Subjective	Auto	J48	97.08±1.09	0.98			0.94±0.02

All features provided the best results for both datasets

Predefined slightly better for ELOC and worse for Subjective

Automatic features acquisition

WEKA WrapperSubsetEval (classifier: J48) and the BestFirst method (selection based on Accuracy and RMSE, five folds, threshold = 0.01).

ELOC, All	ELOC, Predefined	ELOC, Auto	Subjective, All	Subjective, Predefined	Subjective, Auto
Brackets	Brackets	Freq. of ”*”	Assignment	Assignment	Freq. of ”*”
Comments	Comments	Freq. of ”(”	Freq. of ”*”	Comments	Freq. of ”available”
Semicolons	Full length	Freq. of ”;”	Freq. of ”available”	If	Freq. of ”:”
Full length	Semicolons	Freq. of ”/”	Freq. of ”:”	While	Freq. of ”=”
		Full length	Freq. of ”has”	Full length	Freq. of ”has”
			Freq. of ”implied”	Length	Freq. of ”implied”
			Freq. of ”license”	Semicolons	Freq. of ”license”
			Freq. of ”none”	Tokens	Freq. of ”none”
			Freq. of ”reserved”		Freq. of ”reserved”
			Freq. of ”return”		Freq. of ”return”
			Freq. of ”see”		Freq. of ”see”
			Freq. of ”software”		Freq. of ”software”
			Full length		Full length
			Length		Length
			Tokens		Tokens

Results

RQ3: How the choice of classification algorithm affects the classification quality?

Results

Dataset	Features set	Classifier	Accuracy %	Precision	Recall	F-Measure	MCC
ELOC	All	PART	99.55±0.45	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	All	JRip	99.53±0.47	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	All	J48	99.60±0.41	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	Predefined	PART	99.53±0.46	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	Predefined	[illegible]	99.56±0.46	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
ELOC	Predefined	[illegible]	99.60±0.41	1.00±0.01	1.00±0.00	1.00±0.00	0.99±0.01
[illegible]	[illegible]	[illegible]	99.38±0.47	1.00±0.01	0.99±0.01	0.99±0.01	0.99±0.01
[illegible]	[illegible]	[illegible]	99.28±0.47	1.00±0.01	0.99±0.01	0.99±0.01	0.98±0.01
[illegible]	[illegible]	[illegible]	99.18±0.54	1.00±0.01	0.99±0.01	0.99±0.01	0.98±0.01
[illegible]	[illegible]	[illegible]	97.34±1.14	0.98±0.01	0.97±0.02	0.97±0.01	0.95±0.02
[illegible]	[illegible]	[illegible]	96.54±1.20	0.98±0.01	0.95±0.02	0.97±0.01	0.93±0.02
[illegible]	[illegible]	[illegible]	97.18±1.07	0.98±0.01	0.97±0.02	0.97±0.01	0.94±0.02
Subjective	Predefined	PART	95.05±1.45	0.97±0.02	0.93±0.02	0.95±0.01	0.90±0.03
Subjective	Predefined	JRip	95.32±1.44	0.97±0.02	0.93±0.02	0.95±0.02	0.91±0.03
Subjective	Predefined	J48	95.10±1.42	0.97±0.02	0.94±0.02	0.95±0.01	0.90±0.03
Subjective	Auto	PART	97.33±1.08	0.98±0.01	0.97±0.02	0.97±0.01	0.95±0.02
Subjective	Auto	JRip	96.38±1.14	0.98±0.01	0.95±0.02	0.96±0.01	0.93±0.02
Subjective	Auto	J48	97.08±1.09	0.98±0.01	0.96±0.02	0.97±0.01	0.94±0.02

Nearly no differences between the selected ones

PART >? J48 >? JRip

Limitations & hard cases

- Block comments
- Multiple meaningful lines of code in one line
- A single meaningful line in many lines

Questions