A Metric for Software Readability

Raymond P.L. Buse and Westley R. Weimer

{buse,weimer}@cs.virginia.edu

Presenter: Rag Mayur Chevuri

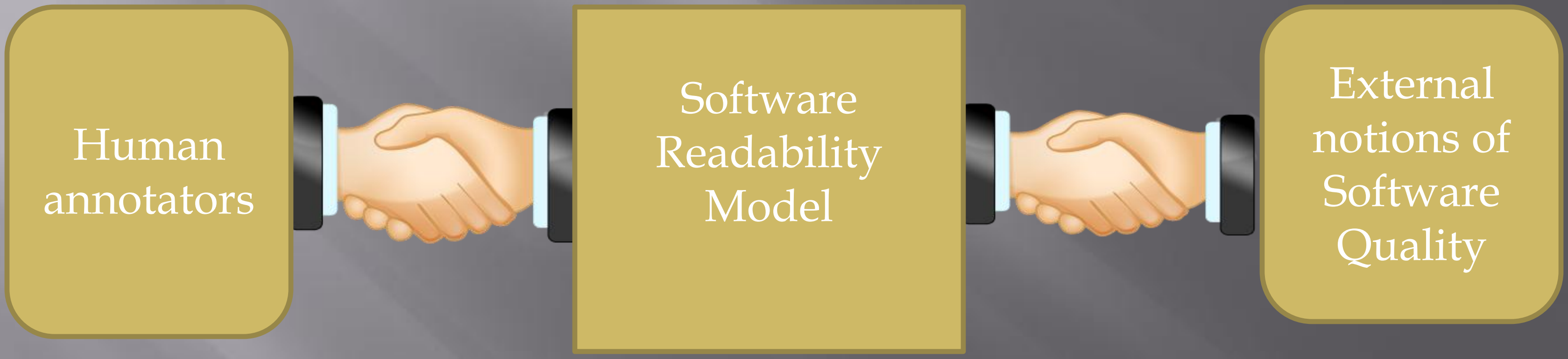
Maintenance consumes 70% of total life cycle cost

Reading code

Most time-consuming component of maintenance

Readability:
Human Judgment of how easy a text is to understand

Our Goal



Human annotators

Software Readability Model

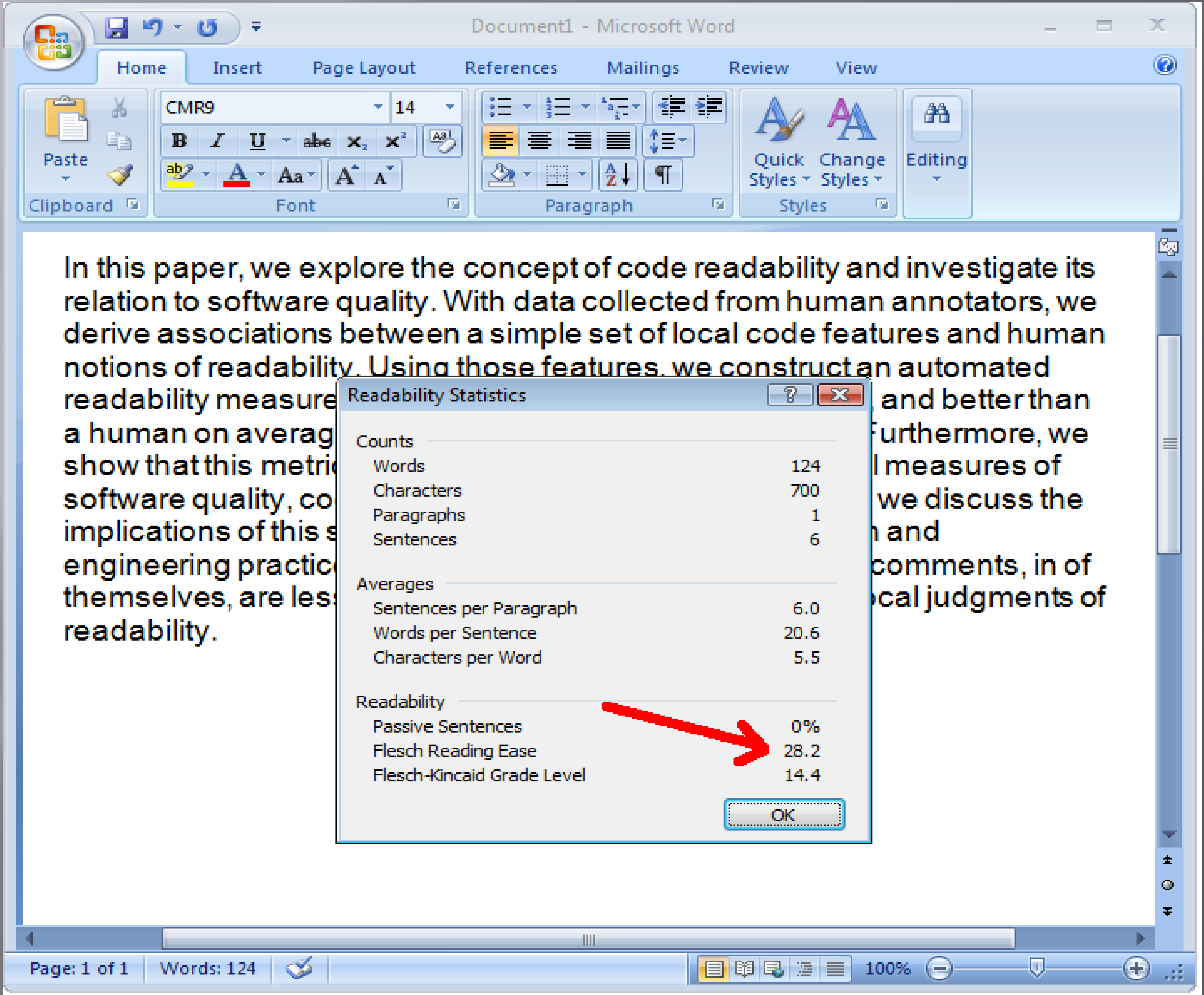
External notions of Software Quality

Readability in Natural Languages

Readability metrics for ordinary text.

- ❖ The Flesch-Kincaid Grade Level. *R. F. Flesch. 1948.*
- ❖ The Gunning-Fog Index. *R. Gunning, 1952*
- ❖ The SMOG Index. *G. H. McLaughlin 1969.*
- ❖ The Automated Readability Index. *J. P. Kinciad, 1970.*

- Based on simple factors like average syllables per word, average sentence length, etc.
- Used with Text Editors like Microsoft word



In this paper, we explore the concept of code readability and investigate its
relation to software quality. With data collected from human annotators, we
derive associations between a simple set of local code features and human
notions of readability. Using those features, we construct an automated
readability measure … and better than
a human on averag … urthermore, we
show that this metri … l measures of
software quality, co … we discuss the
implications of this s … n and
engineering practic … comments, in of
themselves, are less … cal judgments of
readability.

Readability Statistics

Counts	
Words	124
Characters	700
Paragraphs	1
Sentences	6
Averages	
Sentences per Paragraph	6.0
Words per Sentence	20.6
Characters per Word	5.5
Readability	
Passive Sentences	0%
Flesch Reading Ease	28.2
Flesch-Kincaid Grade Level	14.4

Why , the automated readability metric?

- Saves time and cost of Maintenance
- More readable software
- Monitoring and maintaining readability
- Assist inspections.
- Contributes to the overall Software Quality

and Also,

Serve as a requirement for acceptance!!

Contributions

- **A software readability metric, based on local features that**
  - correlates strongly with *human judgments*
  - correlates strongly with *software quality*
- **A survey of 120 human readability annotators**
- **A discussion of the features of metric and its relation to notions of SW quality like defect prediction and code changes**

Approach

- Investigate to what extent the study group agrees on readability.
- Determine a small set of features sufficient to capture the notion of readability for the majority of Annotators
- Discuss correlation between our readability metrics and external notions of software quality

Approach(Contd…)

- Possibilities for extension
- Conclusion.

Complexity metric ≠ Readability metric

•Accidental property	•Essential property
•Arises from system requirements	•Can be addressed easily.
•Based on Sizes of classes, methods ,Extent of Interactions	•Based on local, line-by-line factors
•Not much related to what makes the code understandable	•Judgments of human annotators not familiar with purpose of system.

Human Readability Annotation

- The Experiment:

  Extract large number of readability judgments over short code selections *,the Snippets.*

120 annotators (all are CS students)

- 17 from 100 level courses
- 53 from 200 level courses
- 30 from 400 level courses
- 10 were Graduate students

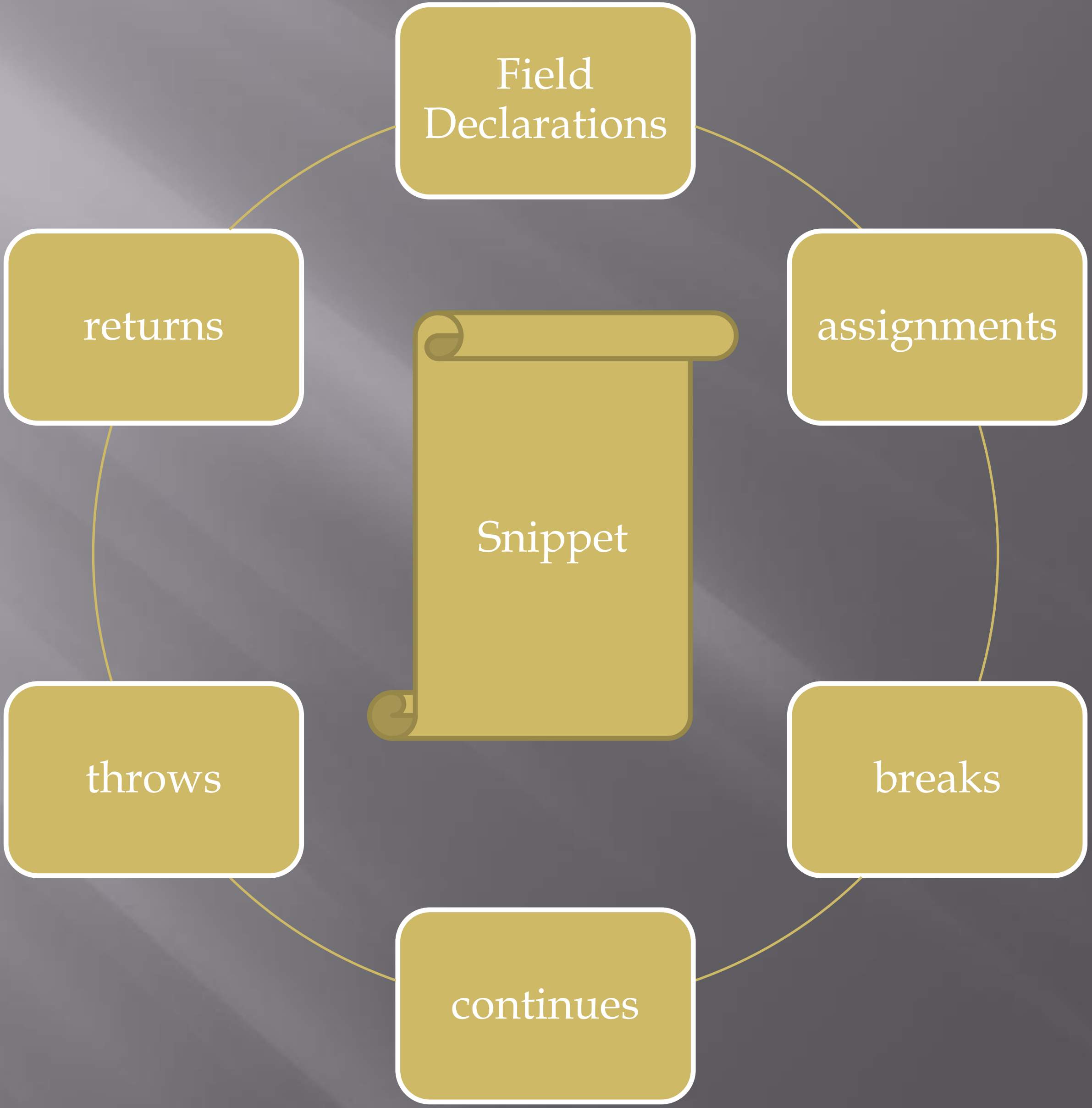
Each annotator scores 100 snippets.

Mapping : Code sample->Finite score domain

Snippet Selection

- Short (but not too short!)
- Logically coherent
- Include adjacent comments
- Not cross scope boundaries

Main focus: Low-level details of readability



Field Declarations

assignments

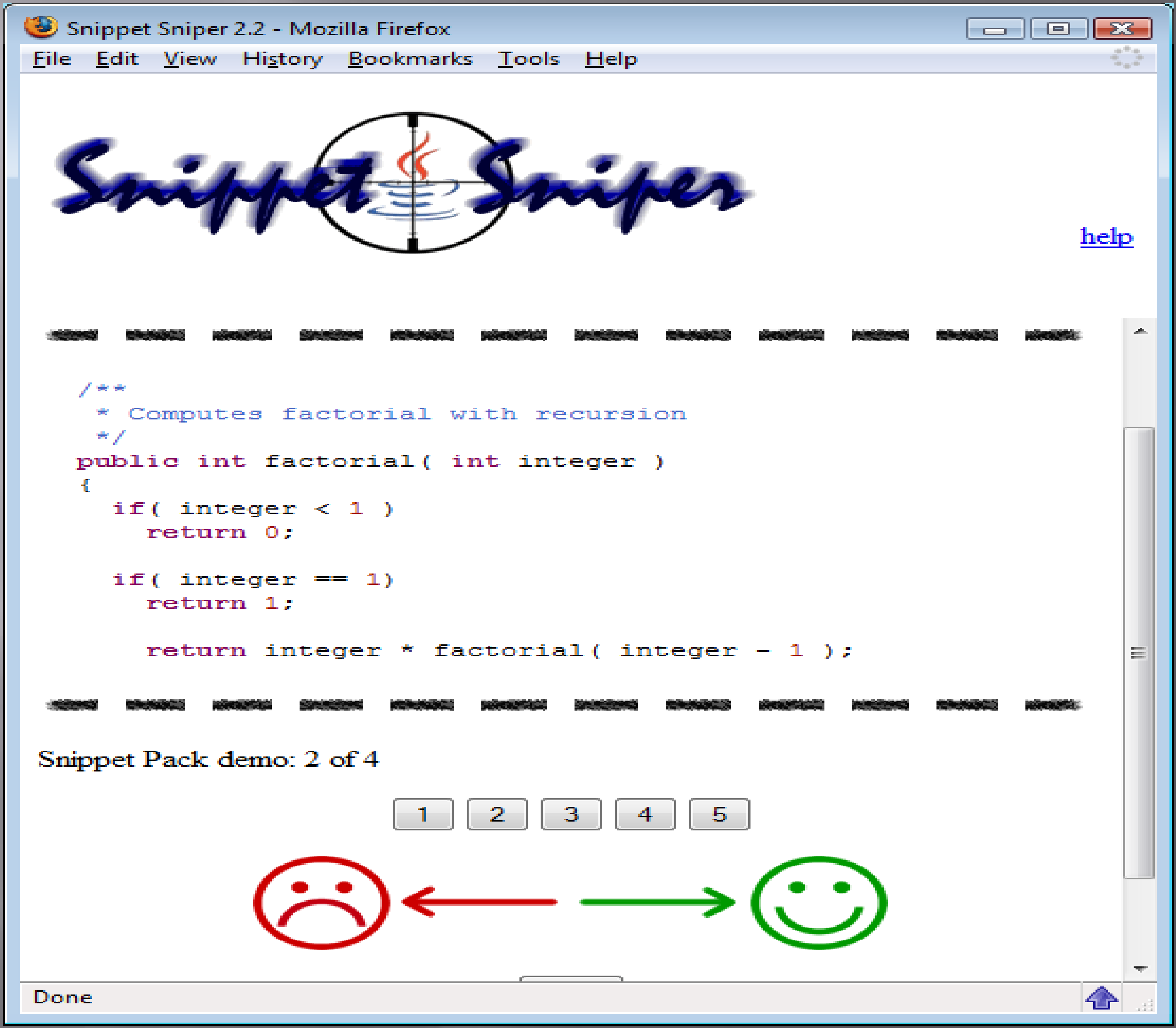
breaks

continues

throws

returns

Snippet

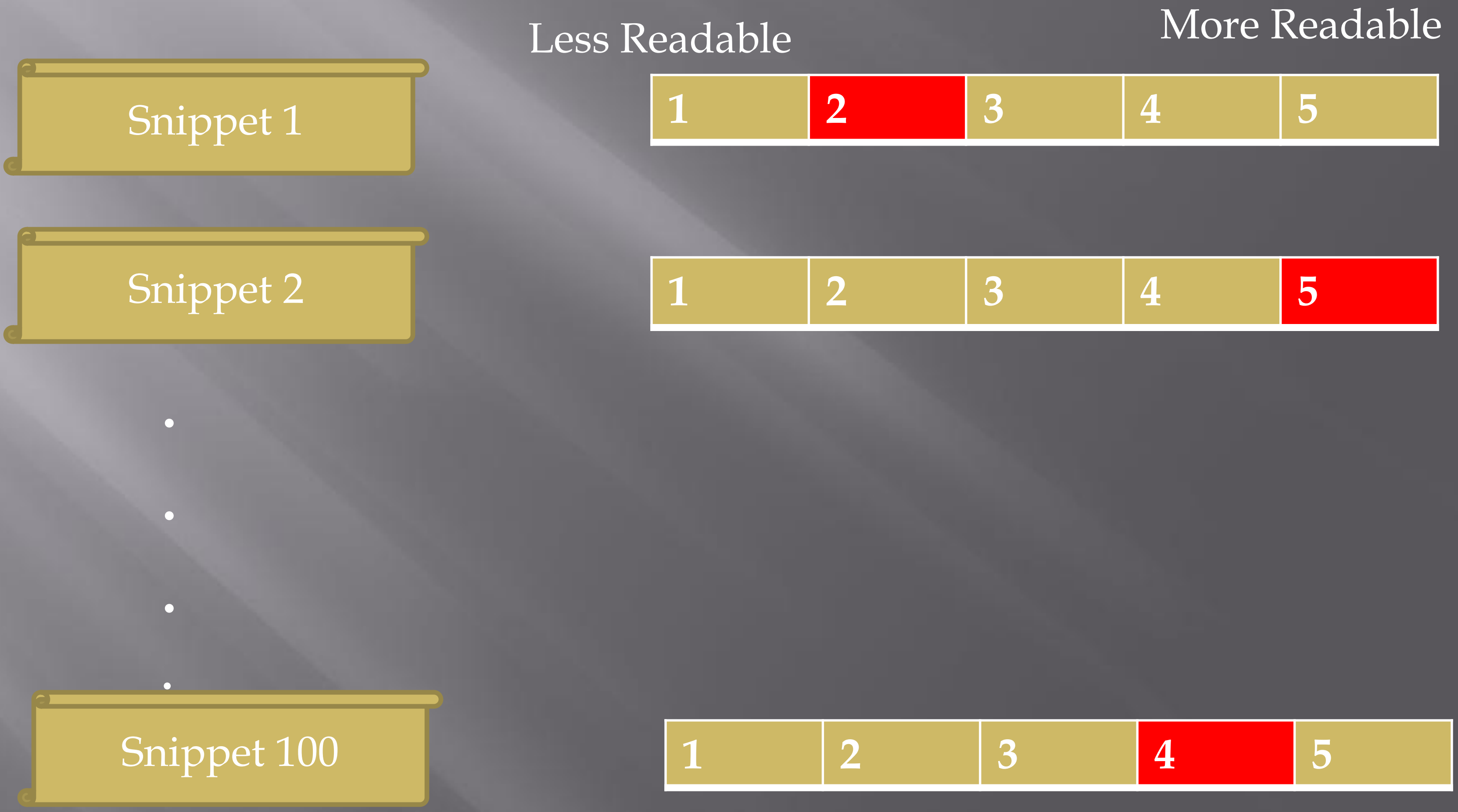


```
/**
 * Computes factorial with recursion
 */
public int factorial( int integer )
{
  if( integer < 1 )
    return 0;

  if( integer == 1)
    return 1;

    return integer * factorial( integer - 1 );
```

Snippet Pack demo: 2 of 4

The scoring begins..

- Tool: Snippet Sniper:



Less Readable

More Readable

Snippet 1

1 [✓ 2] 3 4 5

Snippet 2

1 2 3 4 [✓ 5]

.

.

.

.

Snippet 100

1 2 3 [✓ 4] 5

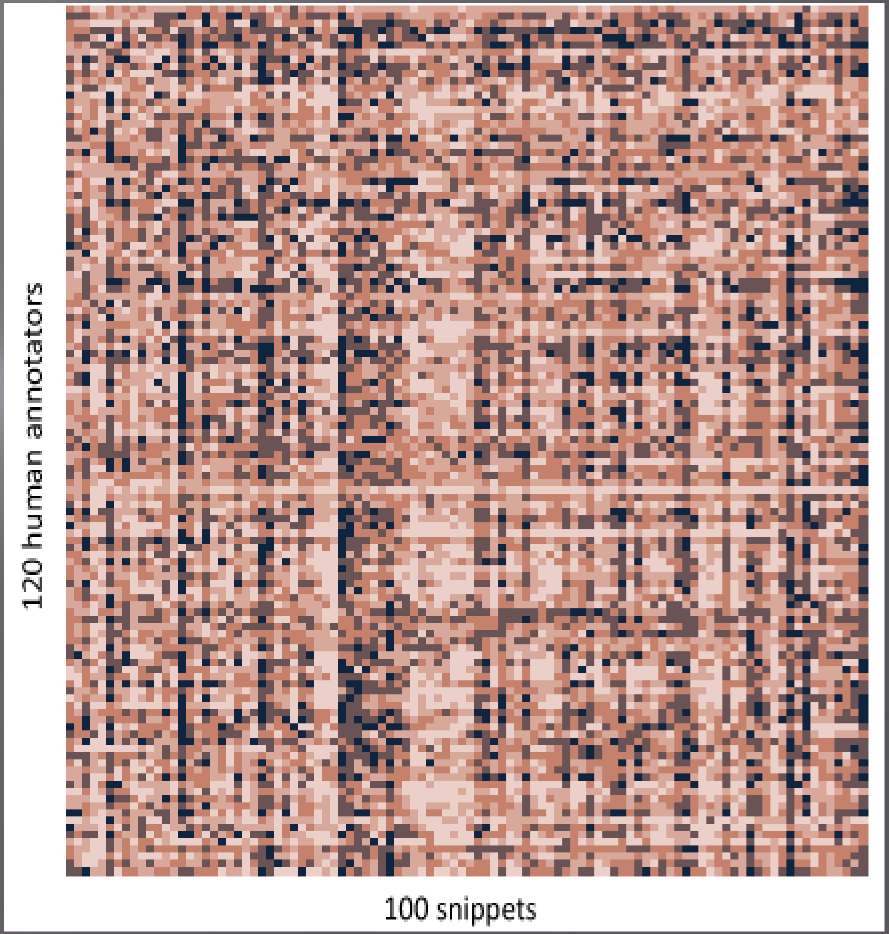
Results

Each box: Judgment of a snippet

Darker colors: Lower readability scores

Lighter colors: Higher readability scores

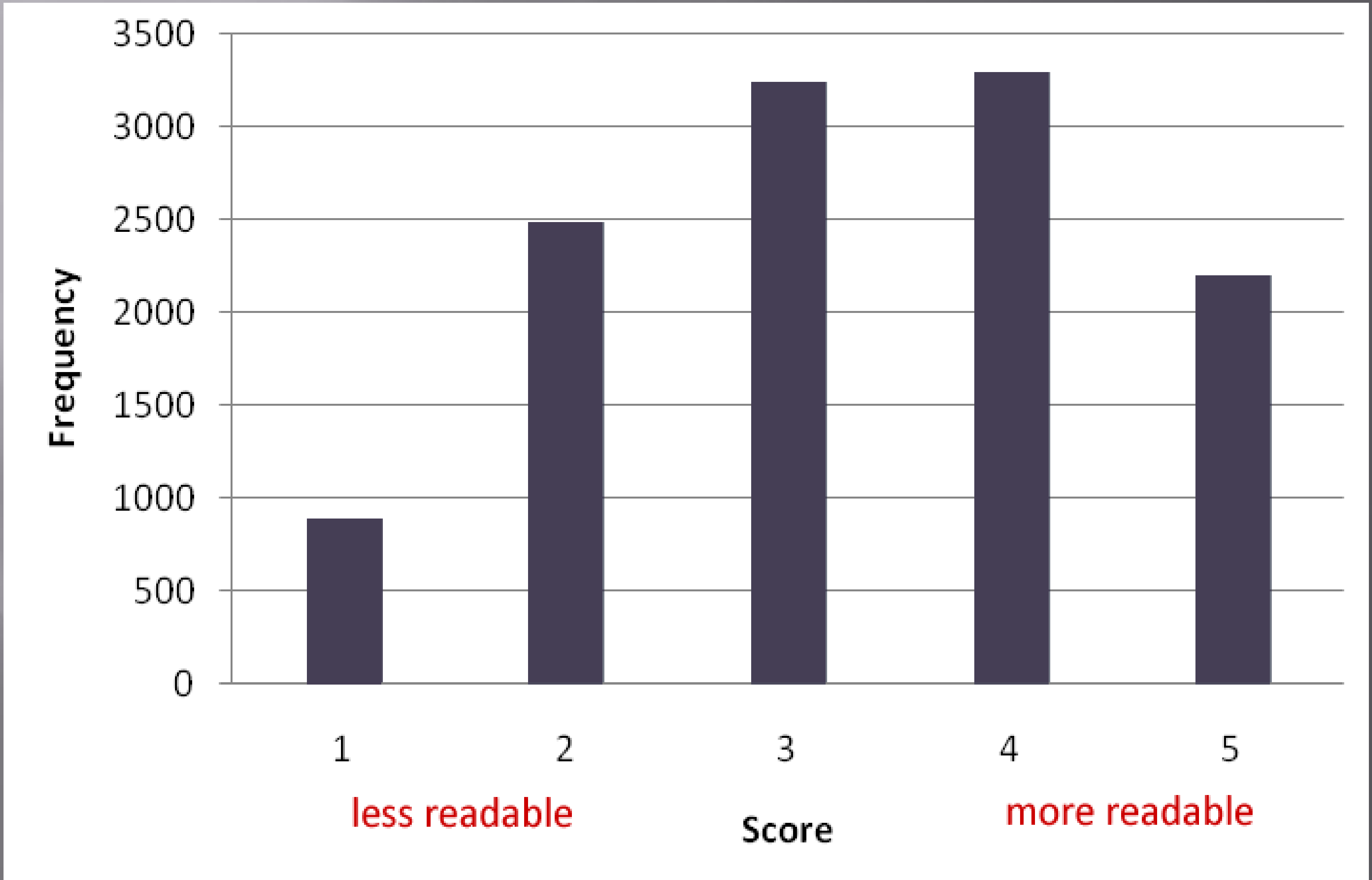
Vertical bands means that they ere judged similarly by many annotators



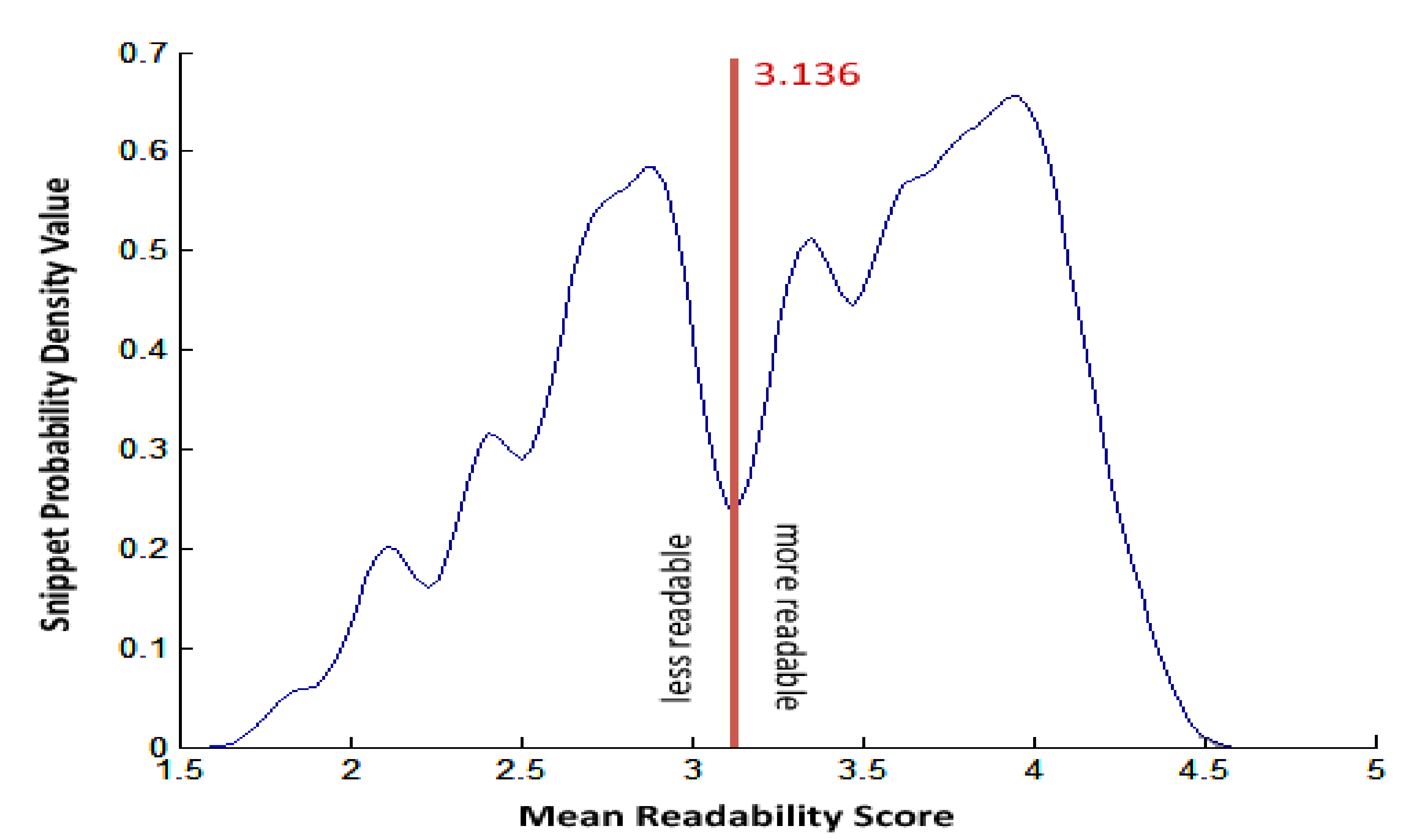
Inter annotator agreement

- To evaluate whether we can extract a single coherent model from this data set.
- Correlation statistic used: Pearson product-moment correlation coefficient
- Compares judgments of annotators two at a time.
- 1–perfect correlation
- 0-no correlation
- Combine large set of judgments into single model by averaging
- Average of 0.56 is considered “Moderate to strong”

Results


3500
3000
2500
2000
1500
1000
500
0
Frequency
1
2
3
4
5
less readable
Score
more readable

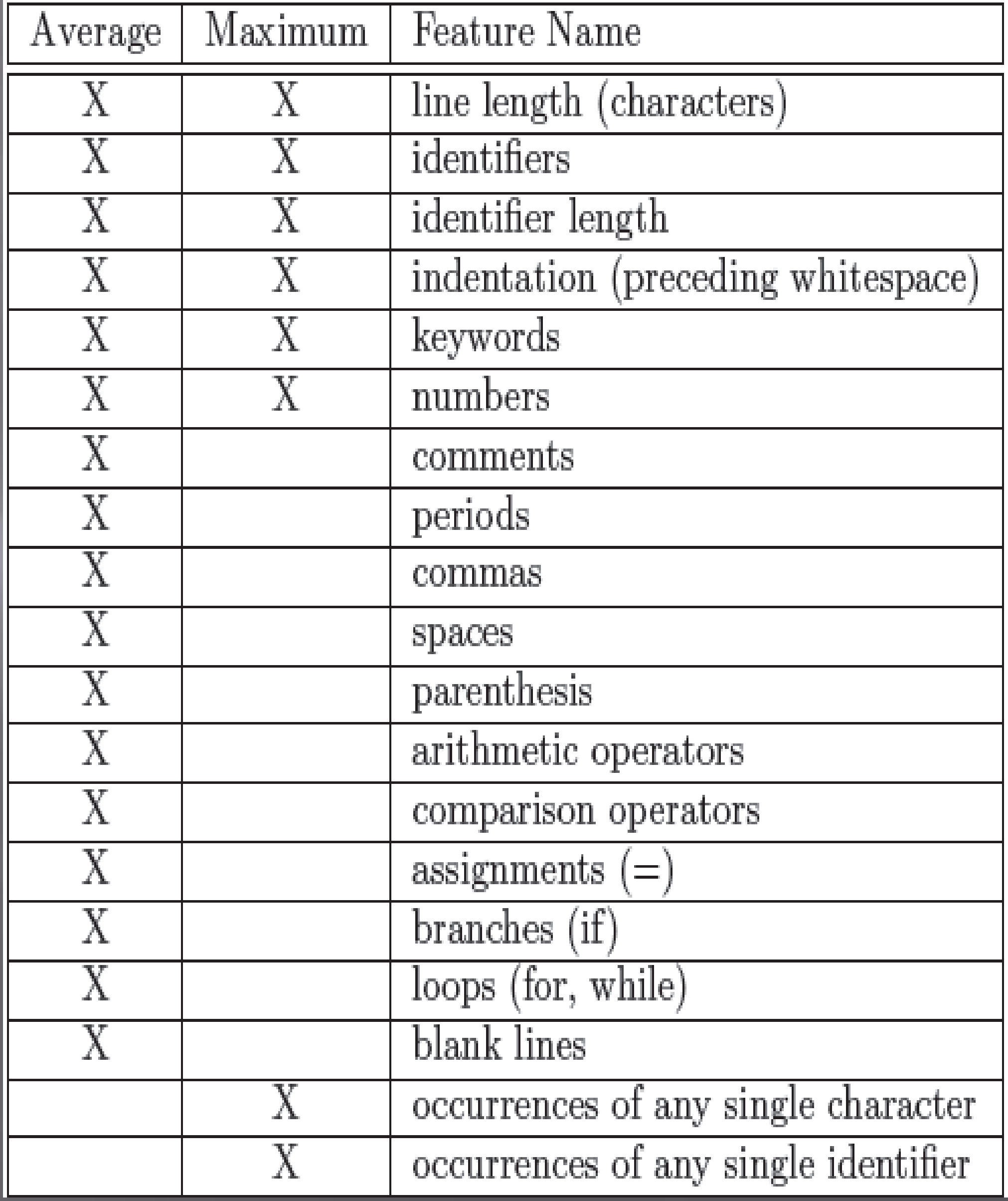
Distribution of average readability of scores across all snippets



Model Generation

We form the set of features that can be statically obtained from a snippet or other block of code

Simple Features related to : Structure, Density, logical complexity, documentation

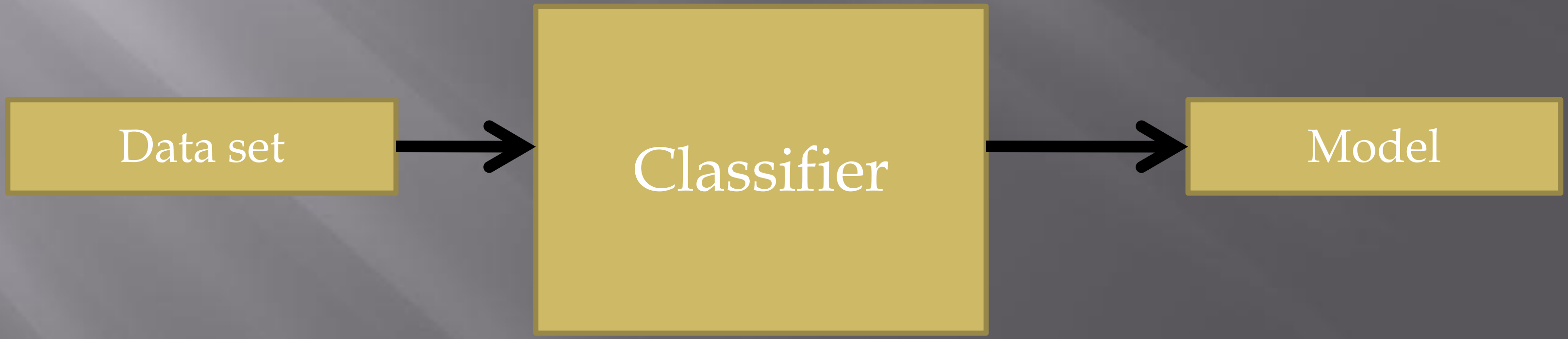


Average	Maximum	Feature Name
X	X	line length (characters)
X	X	identifiers
X	X	identifier length
X	X	indentation (preceding whitespace)
X	X	keywords
X	X	numbers
X		comments
X		periods
X		commas
X		spaces
X		parenthesis
X		arithmetic operators
X		comparison operators
X		assignments (=)
X		branches (if)
X		loops (for, while)
X		blank lines
	X	occurrences of any single character
	X	occurrences of any single identifier

Overview of Machine Learning methods

- Instance=<attribute1,attribute2,…..>
- Dataset-Collection of instances

Training set	**Test set**



Data set

Classifier

Model

Model Generation(Contd..)

- Machine language algorithms
- *Classifier* operating on *Instances*
- *Instance here is a feature vector <Feature1,Feature2,…>*
- *Classifier is given a set of instances+ "correct answer"*
  - *Correct answer:*

<3.14	**>3.14**
Less readable	More Readable

Model Generation(contd..)

- After the training is completed…
- When a new instance is given(that has not been seen before) the classifier is applied and the label is found out i.e, whether it is more readable or less
- Tool: WEKA machine learning toolbox

Available at *http://prdownloads.sourceforge.net/weka/weka-3-6-1jre.exe*

- 10 -fold cross validation .

Model Performance

- Recall(R)=[(#of “more readable” snippets as judged by annotators)/(#of “more readable” snippets as classified by model)]*100%
- Precision(P)=Fraction of snippets classified as “more readable” by the model that were also judged as “more readable” by the annotators.
- f-measure=Combination of R,P.

  Reflects the accuracy of a classifier with respect to more readable snippets

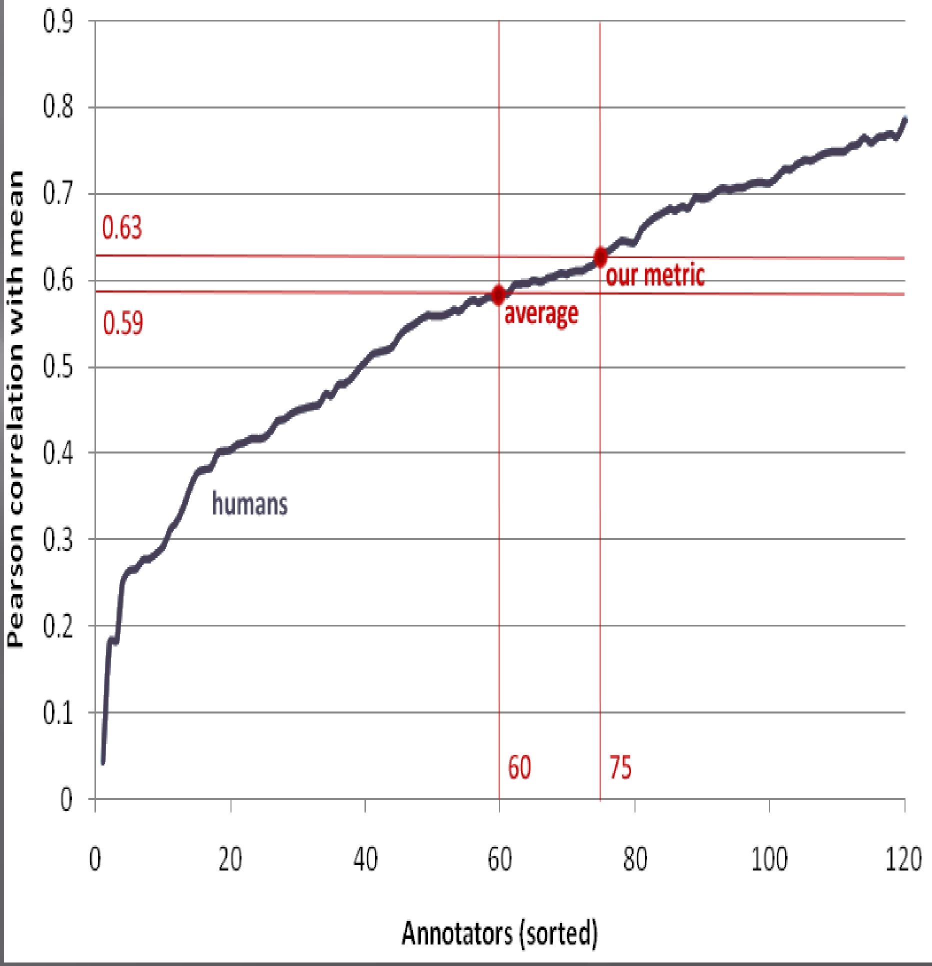
Model Performance (Contd..)

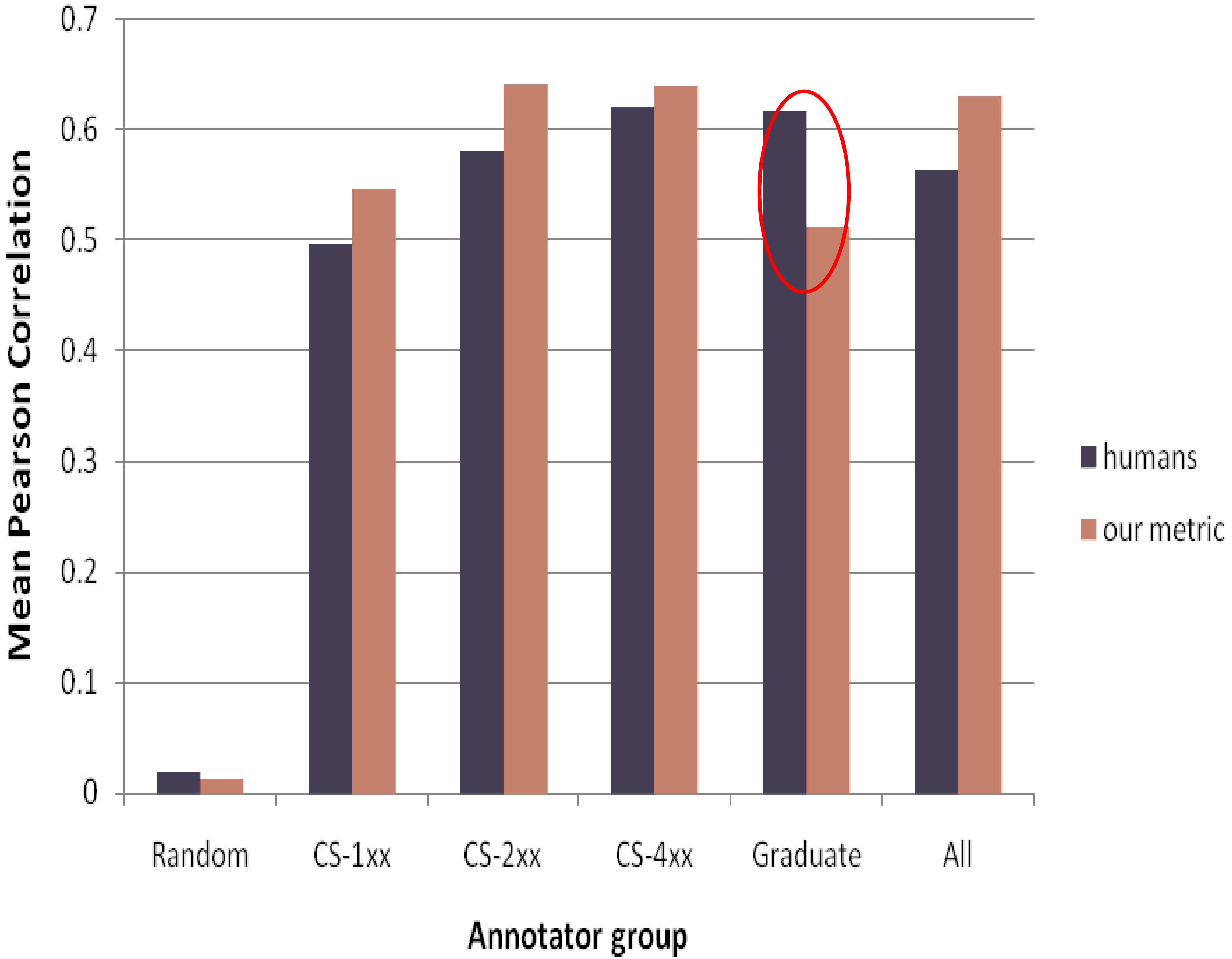
·F-measure when each classifier trained on set of snippets with randomly generated score labels =0.67

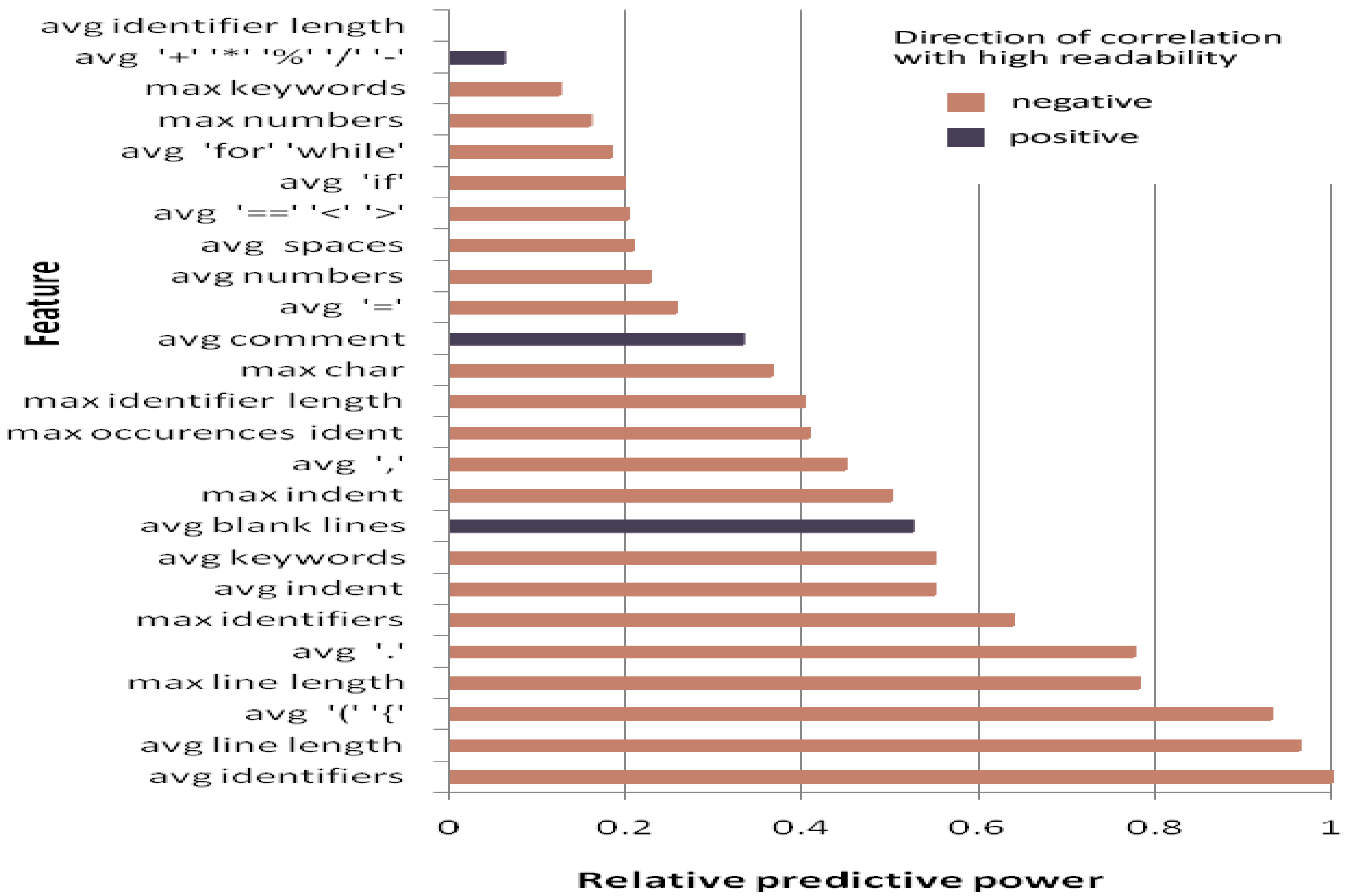
·F-measure when trained on human data =0.8

·Pearson correlation=0.63 when we compare output of Bayesian classifier to the average human score model we trained against.

·Observation: This level of agreement is better than average human in our study produced




Mean Pearson Correlation
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Random
CS-1xx
CS-2xx
CS-4xx
Graduate
All
Annotator group
humans
our metric


Direction of correlation
with high readability
negative
positive
Feature
avg identifier length
avg '+' '*' '%' '/' '-'
max keywords
max numbers
avg 'for' 'while'
avg 'if'
avg '==' '<' '>'
avg spaces
avg numbers
avg '='
avg comment
max char
max identifier length
max occurences ident
avg ','
max indent
avg blank lines
avg keywords
avg indent
max identifiers
avg '.'
max line length
avg '(' '{'
avg line length
avg identifiers
0
0.2
0.4
0.6
0.8
1
Relative predictive power

Correlating Readability with Software Quality

- So far, we constructed an automated model of readability that mimics human judgments
- Now, investigate whether our model of readability compares favorably with external conventional metrics of software quality.
- Find correlation between readability and *FindBugs*(a popular static bug-finding tool)
- Look for similar correlation with changes to code between versions of several large open-source projects

Benchmarks for the experiments

- Open source java projects
- Maturity is self reported:

1-planning

2-pre-alpha

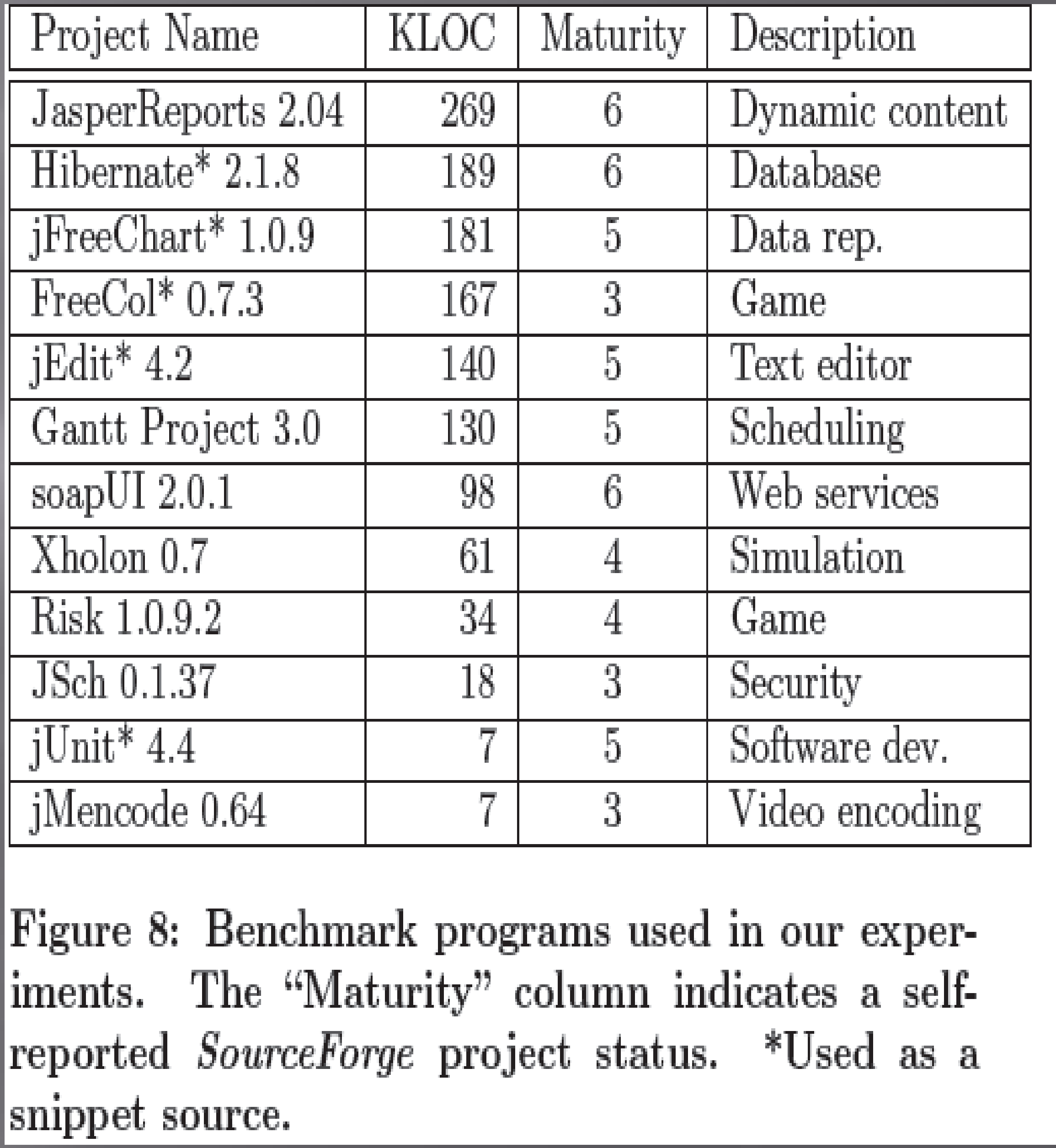
3-alpha

4-beta

5-production/stable

6-mature

7-inactive



Project Name	KLOC	Maturity	Description
JasperReports 2.04	269	6	Dynamic content
Hibernate* 2.1.8	189	6	Database
jFreeChart* 1.0.9	181	5	Data rep.
FreeCol* 0.7.3	167	3	Game
jEdit* 4.2	140	5	Text editor
Gantt Project 3.0	130	5	Scheduling
soapUI 2.0.1	98	6	Web services
Xholon 0.7	61	4	Simulation
Risk 1.0.9.2	34	4	Game
JSch 0.1.37	18	3	Security
jUnit* 4.4	7	5	Software dev.
jMencode 0.64	7	3	Video encoding

Figure 8: Benchmark programs used in our experiments. The "Maturity" column indicates a self-reported *SourceForge* project status. *Used as a snippet source.

Readability Correlations

- Experiment 1:Correlate defects detected by *Findbugs* with our readability model at the function level

1. Run *FindBugs* on the benchmarks.
2. Extract all functions and partition into two.
3. Run already trained classifier on the set of functions

Readability Correlations(Contd..)

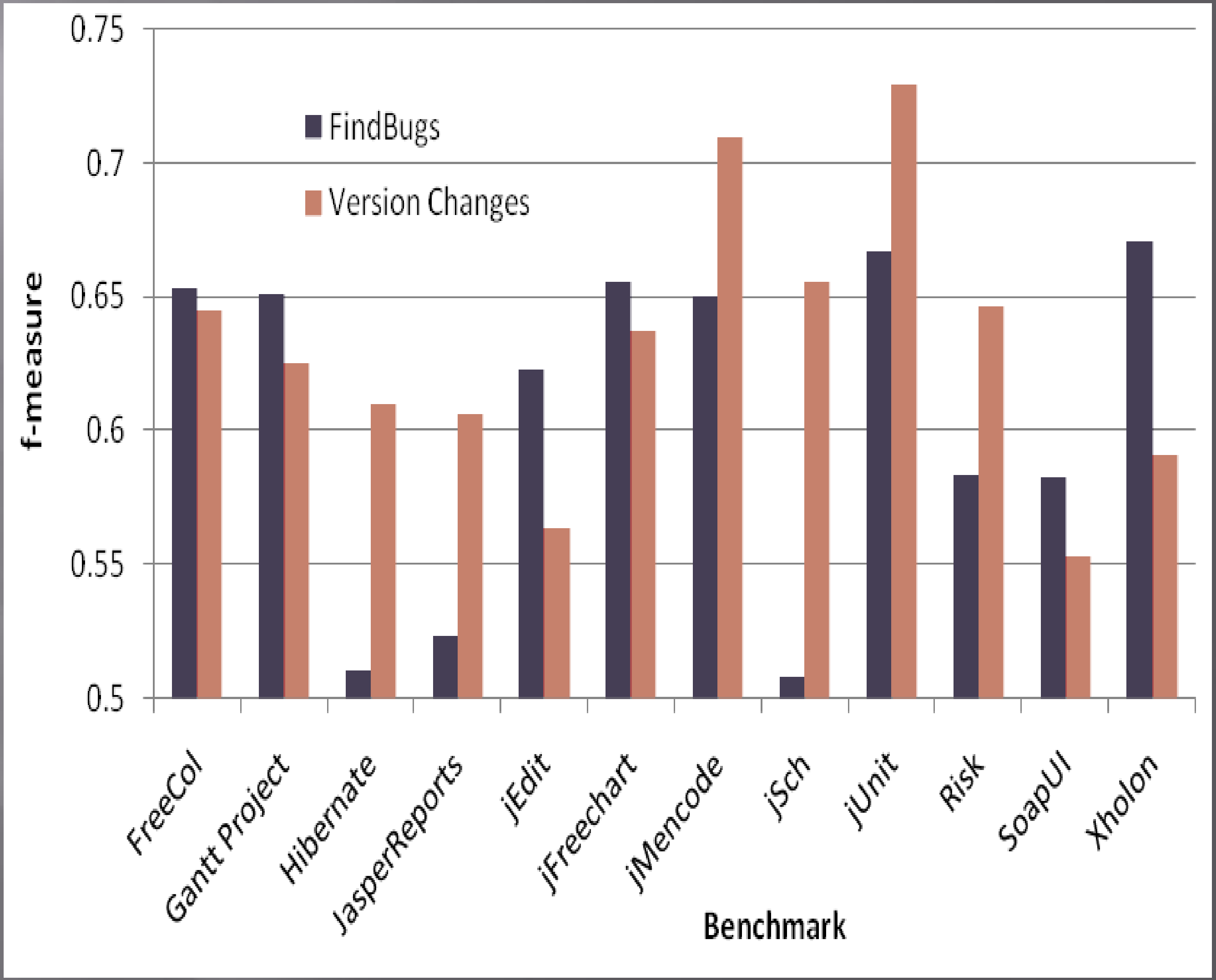
- Experiment 2:Correlate future code churn to readability

1. Use the same set up as first experiment
2. Use readability to predict which functions will change in successive releases.
3. Instead of contains a bug“ we attempt to predict is going to change soon."

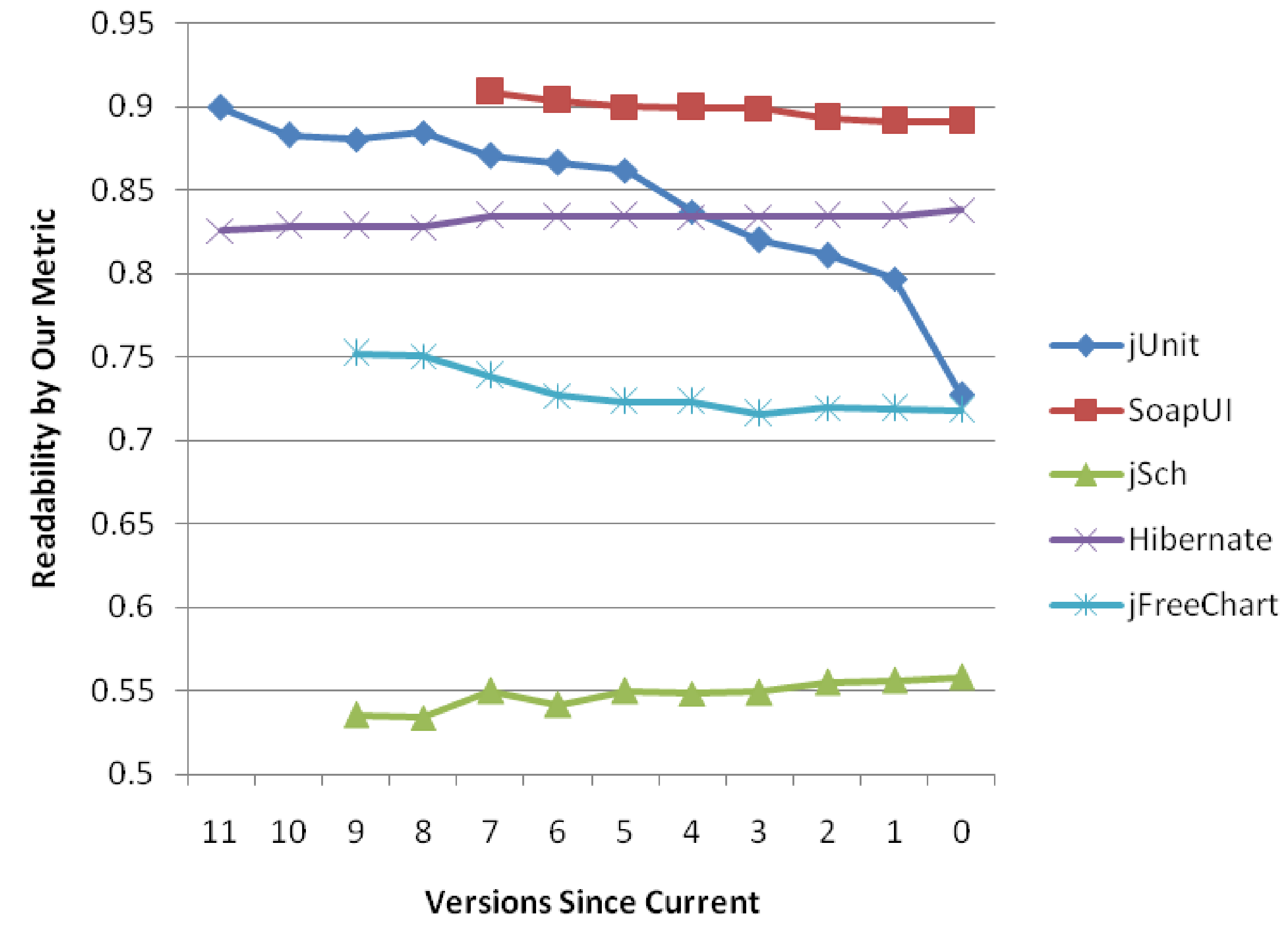
Results

F measure for Experiment 1: 0.63

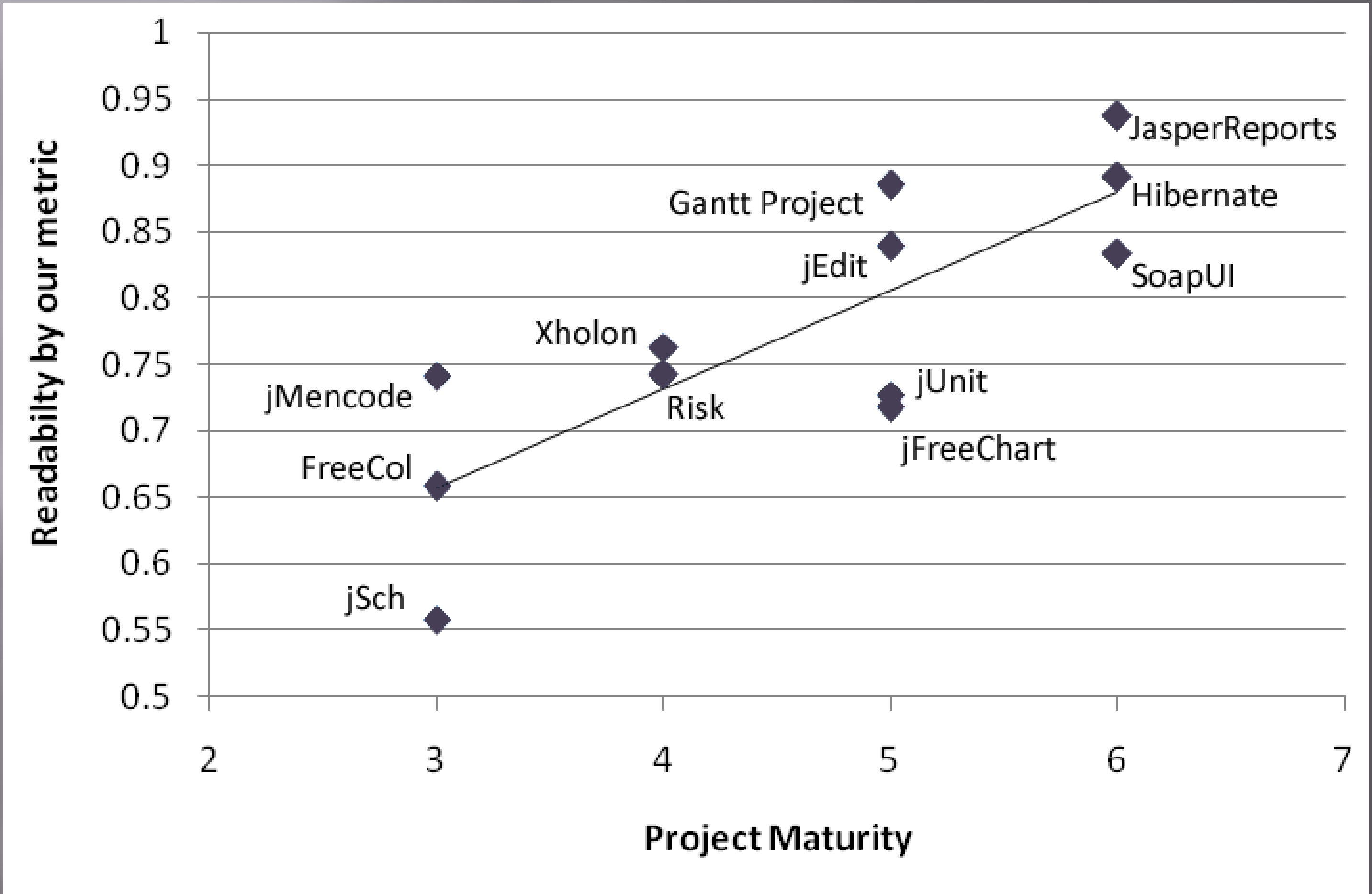
F measure for Experiment 2: 0.5



Software Lifecycle and Readability



Readability vs. Maturity



Some more facts observed in the study.

- Identifier length's influence on readability=NILL
- Descriptive identifiers are sometimes useful and sometimes not
- Comments have only 33% relative power to readability
- # of identifiers and characters have strong influence on our readability metric
- Languages should add additional keywords in order to have fewer new identifiers.

Limitations

- 100 snippets is a very small sample
- All the annotators were students .There were no software professionals or any experienced persons involved in the judgments

Related Work

- Automated readability metrics for Natural languages .*The Flesch-Kincaid Grade Level. R. F. Flesch. 1948*
- PMD and Java coding standards to enforce some code standards. *S. Ambler. Java coding standards., 1997.*
- Machine learning on source code repositories, defect prediction in Software engineering and Programming languages. *Predicting defect densities in source code les with decision tree learners. P. Knab, M. Pinzger, and A. Bernstein., 2006.*

Future work

- Investigate whether personalized model adapted over time ,will be effective in characterizing code reliability
- Consider size of compound statements
- Include Readability measurement tool in IDEs
- Express metric using simple formula with small number of features

Conclusion

- Producing an effective Readability metric from the judgments of annotators.
- May not be a universal model.
- The model considers relatively simple set of low-level code features
- Readability exhibits significant levels of correlation with more conventional metrics of software quality like Defect Reports, Version changes, and Program maturity
- Factors influencing readability might help in Program language design

Thanks