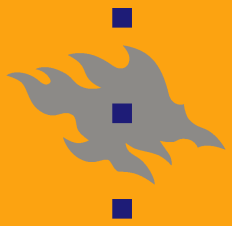
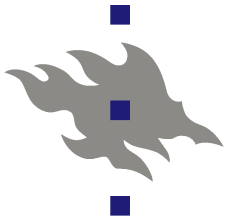


HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Automated tools for source code plagiarism detection

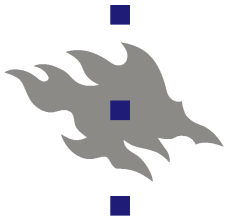
Kristian Wahlroos
April 26, 2017

University of Helsinki
Department of Computer Science



Outline

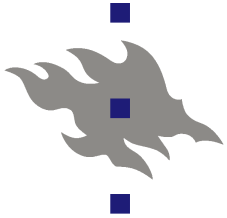
- Introduction
  - Problem
  - Plagiarism
  - Motivation
- Methodology
  - Literature review
- Results
  - Overview
  - Data
  - Methodologies
  - Feature extraction
- Discussion



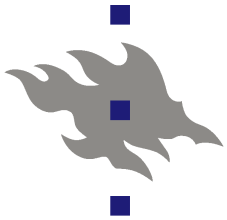
Introduction

- MOOC's have gained popularity in recent years
  - Especially programming related MOOC's[1]
  - Independent assignments
  - No live-presence required
- Number of students often large
- Trust is thus usually one-sided
  - Belief that students do tasks by themselves
  - Not actively monitored
  - Cheating is in form of plagiarism
  - Many potential plagiarism scenarios

[1] http://blog.edx.org/10-most-popular-courses-edx-courses-in-2016

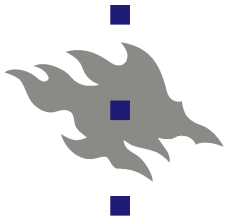


- Source code plagiarism is a problem consisting many forms
  - Straight plagiarism
  - Too intense group work
  - Code sharing
  - Obfuscation
- Lots of students $\rightarrow$ impossible to detect manually in reasonable time
  - Lot of data available
  - Need for automated tools



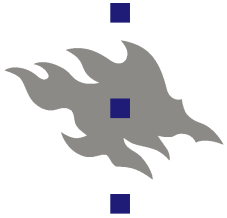
In this study

- Finding a suitable machine learning tool set for detecting source code plagiarism
- Motivated by
  - Could be used in University of Helsinki's course *Introduction to programming*
  - Interesting topic
  - Machine learning methods benefit from a lot of data
- Results reflected to the usage in a academic course

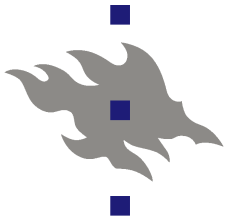


Methodology

- Performing literature review with *Google Scholar*
- Collected 8 papers
- Two-step search process
  - Limit by overall keywords occurrences
  - Limit by title/abstract/keywords
- Keywords
  - Direct matches: **machine learning, plagiarism, code, programming**
  - Non-direct: **authorship, identification**

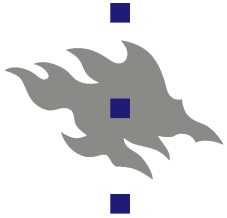


- Limited years starting from 2006
  - Believed to contain more recent programming languages
  - MOOC's are relatively new concept
  - Machine learning methods have changed
- Doing comparison between papers
  - Model accuracy
  - Data
  - Machine learning methodology
  - Feature extraction

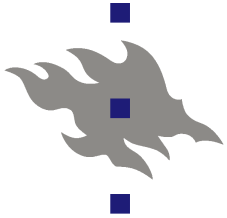


Results

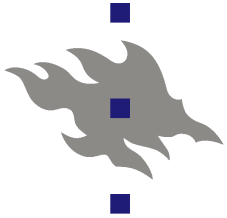
- 8 papers from 2007 to 2015
  1) *A machine learning based tool for source code plagiarism detection*, 2011
  2) *De-anonymizing programmers via code stylometry*, 2015
  3) *Detecting outsourced student programming assignments*, 2008
  4) *Pde4java: Plagiarism detection engine for java source code: a clustering approach*, 2008



5) *A probabilistic approach to source code authorship identification*, 2007
6) *Using code metric histograms and genetic algorithms to perform author identification for software forensics*, 2007
7) *Who wrote this code? Identifying the authors of program binaries*, 2011
8) *An application for plagiarized source code detection based on a parse tree kernel*, 2013



- Studies divide into two categories
  - Attribute counting (4 papers)
  - Structure based (4 papers)
- Attribute counting is easy and fast $\rightarrow$ directly from the source code
- Structure based require parsing
- Model accuracies are reported in two ways
  - Traditional classification accuracy
  - How close the model was to human labeling



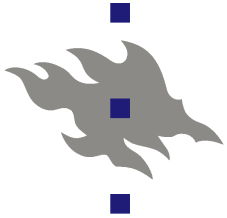
- Accuracies ranged from 69% to over 90%
  - Highest used mixture of stylistic and structural approach
  - E.g. 93% same results compared to human validator
- Plagiarism detection is close to authorship identification
  - Classifying anonymous source code
  - Clustering similar documents together
  - Finding stylistic nuances
  - Trying to capture the logical structure

Data sets in studies

- No clear difference between stylistic studies and structural studies
  - Partially reported data sets
  - Just few authors in stylistic studies

Attr./Paper	1	5	3	6	8	2	4	7
Size	741	200	83	4068	555	N/A	326	203
Authors	10	8	12	20	N/A	1600	N/A	32
Structural	No	No	No	No	Yes	Yes	Yes	Yes

Table: Reported data sets used in papers



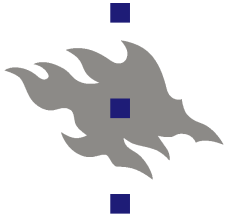
- Data sets are often collected from course assignments
- Open source projects are utilized to gather the data set
- Competitions
  - *Google Code Jam*
  - Explains the large number of possible authors

Machine learning methods

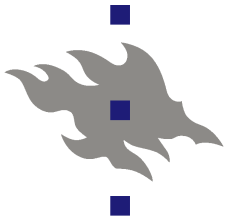
Attr./Paper	1	5	3	6	8	2	4	7
Method	E_3	NB/VFI	DT	GA	PTK	RF	DM	K-M/SVM
Structural	No	No	No	No	Yes	Yes	Yes	Yes

Table: Methods used in studies

E_i	Ensemble of i models
NB	Naive Bayes
VFI	Voting Feature Interval
DT	Decision Tree
GA	Genetic Algorithm
PTK	Parse Tree Kernel
RF	Random Forest
DM	Data Mining (DBSCAN)
K-M	K-means Clustering
SVM	Support Vector Machine

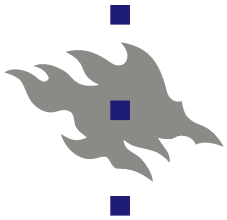


- Many various algorithms used
  - Probabilistic
  - Trees
  - Genetic algorithm
  - Clustering
- Structural studies tend to favor tree-structures and clustering
  - Easy due to nature of code structure
  - Overcomes the obfuscation
  - Pre-parsing often required
- More variance in stylistic studies
  - Probabilistic models
  - Genetic algorithm
  - Decision tree



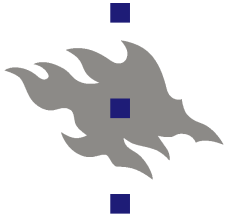
Features

- *How source code is transformed to feature vector?*
- *How similarity is being measured?*
- Inspected from two viewpoints
  - Attribute counting (stylistic)
  - Structural
- Most stylistic studies extract the features directly from the source code
- Structural studies tend to parse the document and use abstract syntax tree as a base for features

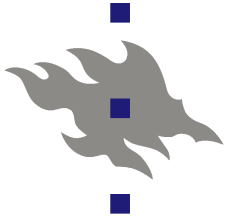


Features - stylistic

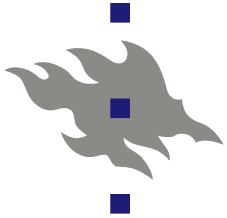
1) *A machine learning based tool for source code plagiarism detection*, 2011
   - Code as set of tokens
   - Parsing with ANTLR-parser
   - 9 different metrics
     - Number of characters per line
     - Number of words in document
     - Number of access modifiers



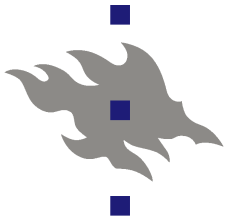
5) *A probabilistic approach to source code authorship identification*, 2007
   - Three step pipeline
     I) Extract metrics
     II) Filter metrics
     III) Form writer profiles
   - Style and pattern based metrics
     - Style consists line sizes, leading spaces and commas as distributions
     - Pattern based is distribution of possible N-grams
   - 4 was considered as the best length in N-gram



3) *Detecting outsourced student programming assignments*, 2008
   - Programmer profiles (writer profiles)
   - Utilizing *Weka* to parse documents
   - Six metrics
     - zipped size
     - lines of code
     - number of variables
     - number of for-loops
     - number of comments
     - variable lengths



6) *Using code metric histograms and genetic algorithms to perform author identification for software forensics*, 2007
   - 18 base metrics
   - Gathered metric values are transformed into distributions
   - Text-based and syntactic metrics
   - Genetic algorithm to find the best combination
     - usage of curly braces
     - comment amount and style
     - indentation
     - inline spaces
     - length of lines
     - switch-cases
     - control flows (if-else)
     - frequencies of first characters in variables



Features - structural

8) *An application for plagiarized source code detection based on a parse tree kernel*, 2013
   - Parse tree kernel
     - Produces similarities between source codes
     - Tree kernels are used often in linguistics
   - Similarity matrix to group source codes
   - Utilizing heavily the tree structure

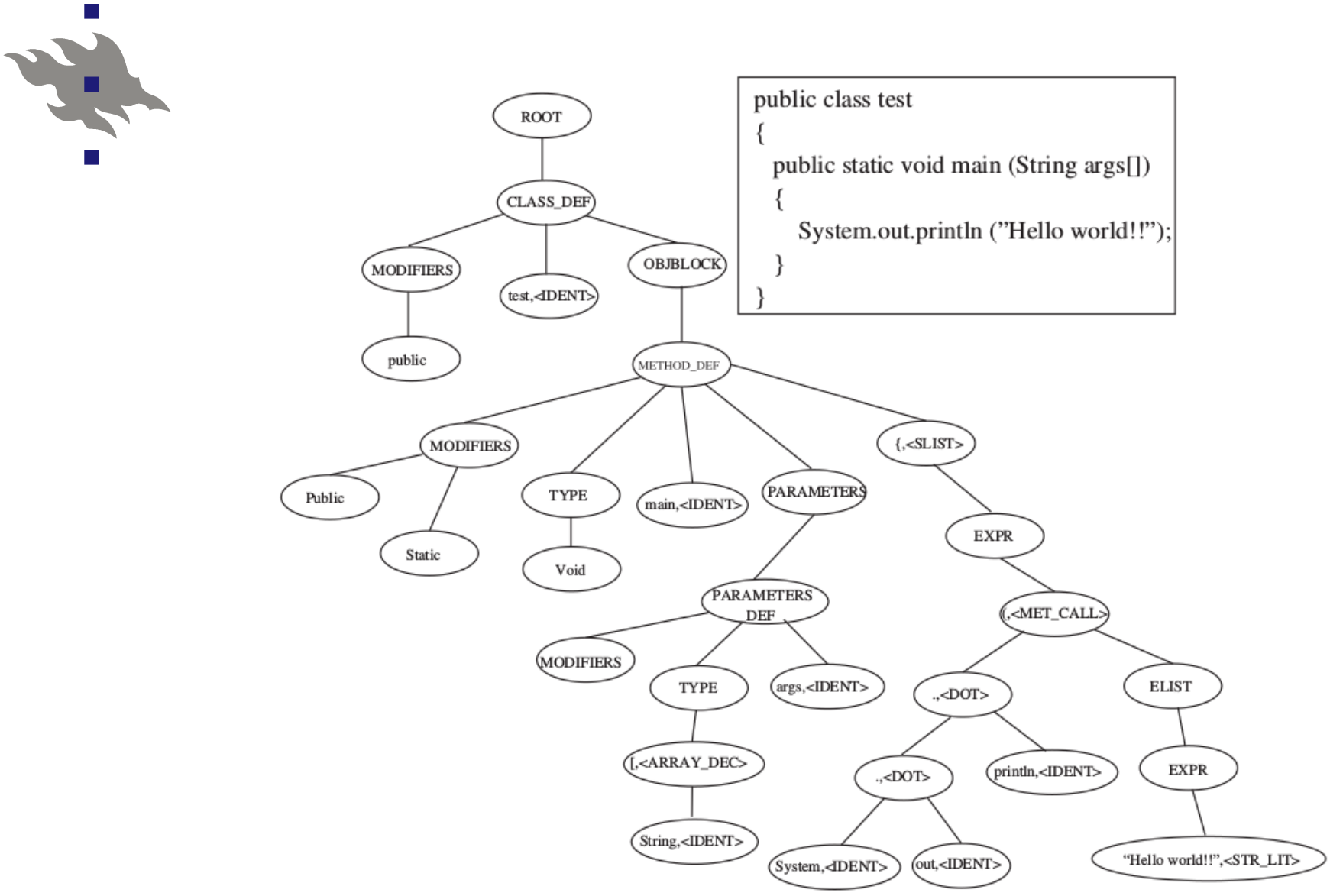
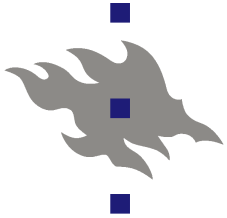
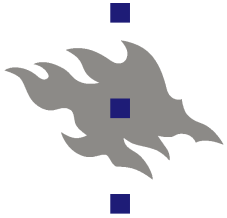


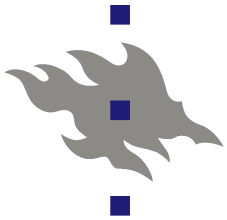
Figure: Example of parse tree in study 8



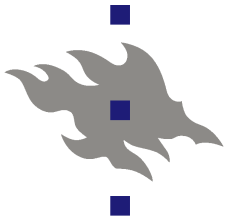
2) *De-anonymizing programmers via code stylometry*, 2015
   - Building *Code Stylometry Feature Set*
   - Utilizing abstract syntax tree (AST)
   - 3 classes of features used
     - Lexical (number of functions/keywords)
     - Layout (indentation, whitespace)
     - Syntactic (maximum depth of AST, AST bigrams)



4) *Pde4java: Plagiarism detection engine for java source code: a clustering approach*, 2008
   - Source code into n-grams via tokenization
   - 4-grams
   - Similarity measure by *Jaccard coefficient*
     - In simplified form: ratio of specific n-gram between two documents
   - Similarity is used to group documents into clusters

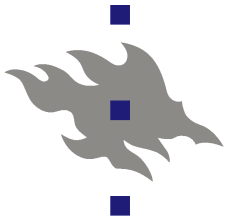


7) *Who wrote this code? Identifying the authors of program binaries*, 2011
   - Graphs and N-grams as features
   - Binary code first into machine language
   - Flow graphs
     - Idioms: short instruction sequences
     - Graphlets: local structure of the program
     - Supergraphlets: combine graphlets and simplify
     - Call graphlets: interaction between external libraries
   - N-grams size of 4-bytes is used



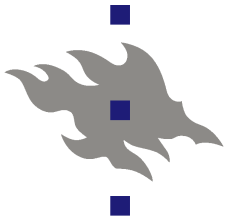
Recap

- Two distinctions between studies: stylistic and structural
  - Stylistic is can be gained directly from the source code
  - Structure deals with tree-like data
- Data is quite easily obtainable
  - Open-source projects
  - Competitions
  - Courses inside universities
- It all comes down to author identification
  - *Who wrote this code?*



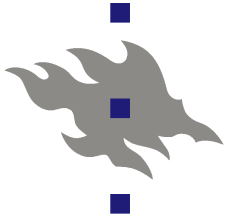
Criticism

- Some papers are not reporting explicitly the data and author counts
- No justification/comparing between different machine learning models
- Detection model is rarely put into real life test
  - No proof of the actual (real-life) performance
  - Plagiarism accusation is difficult topic
  - What to do if task is very limited



Reflection - *Introduction to Programming*

- Six weeks of assignments 1 machine test
- Based on study 2
  - 3 classes of features: lexical, stylistic and syntactic
  - Classifying 250 programmers should give good quite accurate results
- Training the model (programmer profiles) for six weeks
- Testing against unseen data $\rightarrow$ machine test
- If model would alarm, requires manual step



Thank you